# STRIPE: Remote Driving Using Limited Image Data

Jennifer S. Kay
January 1997
CMU-CS-97-100



School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15217

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy*



Thesis Committee:
Chuck Thorpe, Chair
Bonnie John
Eric Krotkov
Larry Matthies, JPL



Jennifer Kay was funded by a NASA Graduate Student Researchers Program Fellowship grant number NGT-51292. This research was also partly sponsored by DARPA, under contracts "Technology Enhancements for Unmanned Ground Vehicles," contract number DAAE07-96-C-X075 and "Unmanned Ground Vehicle System" contract number DAAE07-90-C-R059 monitored by TACOM; and "Perception for Outdoor Navigation," contract number DACA76-89-C-0014 monitored by the US Army Topographic Center. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the funding agencies

KEYWORDS: Artificial Intelligence, Human Computer Interaction, Mobile Robots, Semi-Autonomous Vehicles, Unmanned Ground Vehicles, Vehicle Teleoperation, Polyhedral-Earth Reprojection, Low-Bandwidth Teleoperation, High-Latency Teleoperation, Navlab, STRIPE.


School of Computer Science

**DOCTORAL THESIS**
**in the field of**
**Computer Science**

# *STRIPE: Remote Driving Using Limited Image Data*

**JENNIFER KAY**

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

**ACCEPTED:**

Charles E Thorpe — THESIS COMMITTEE CHAIR — Jan 3 1997 — DATE

[illegible] — DEPARTMENT HEAD — 1/8/97 — DATE

**APPROVED:**

R. Rashid — DEAN — 1/28/97 — DATE

*In Memory of Julia Jacqueline Kay*

# Abstract

Driving a vehicle, either directly or remotely, is an inherently visual task. When heavy fog limits visibility, we reduce our car's speed to a slow crawl, even along very familiar roads. In teleoperation systems, an operator's view is limited to images provided by one or more cameras mounted on the remote vehicle. Traditional methods of vehicle teleoperation require that a real time stream of images is transmitted from the vehicle camera to the operator control station, and the operator steers the vehicle accordingly. For this type of teleoperation, the transmission link between the vehicle and operator workstation must be very high bandwidth (because of the high volume of images required) and very low latency (because delayed images can cause operators to steer incorrectly).

In many situations, such a high-bandwidth, low-latency communication link is unavailable or even technically impossible to provide. Supervised TeleRobotics using Incremental Polyhedral Earth geometry, or STRIPE, is a teleoperation system for a robot vehicle that allows a human operator to accurately control the remote vehicle across very low bandwidth communication links, and communication links with large delays.

In STRIPE, a single image from a camera mounted on the vehicle is transmitted to the operator workstation. The operator uses a mouse to pick a series of "waypoints" in the image that define a path that the vehicle should follow. These 2D waypoints are then transmitted back to the vehicle, where they are used to compute the appropriate steering commands while the next image is being transmitted. STRIPE requires no advance knowledge of the terrain to be traversed, and can be used by novice operators with only minimal training.

STRIPE is a unique combination of computer and human control. The computer must determine the 3D world path designated by the 2D waypoints and then accurately control the vehicle over rugged terrain. The human issues involve accurate path selection, and the prevention of disorientation, a common problem across all types of teleoperation systems. STRIPE is the only semi-autonomous teleoperation system that can accurately follow paths designated in monocular images on varying terrain. The thesis describes the STRIPE algorithm for tracking points using the incremental geometry model, insight into the design and redesign of the interface, an analysis of the effects of potential errors, details of the user studies, and hints on how to improve both the algorithm and interface for future designs.

# Acknowledgments

It is difficult to reduce to a few paragraphs the thanks that I owe to all of the people who have helped me make it through my graduate student career. So many people have helped and supported me, and made this such a special place.

For a start, thanks to my advisor, Chuck Thorpe. Besides his amazing skills in robotics, Chuck can recite poetry faster than a speeding bullet, and approximate the tangent of small angles in a single bound. His encouragement, enthusiasm, and guidance kept me "going for it" despite some seemingly insurmountable hurdles. Thanks also my thesis committee: Bonnie John, Eric Krotkov, and Larry Matthies, for all of their comments, suggestions, and help. An extra thanks to Bonnie, for taking the time to help me in the design and analysis of my user study. Thanks to Sharon Burks and Catherine Copetas for their support, advice, ability to hide all of the bureaucracy that graduate school could be from the students, and for knowing the answers to just about any question presented to them.

Thanks to all the members of the Unmanned Ground Vehicle and Automated Highway System projects for all their help. In particular, thanks to Jay Gowdy, Todd Jochem, and Dean Pomerleau for lots of software that made writing my own software significantly easier. Thanks to Martial Hebert, for support in writing code and text. Thanks to Barry Brumitt, Terry Fong, John Hancock, and Dirk Langer for keeping various parts of the vehicle up and running. Thanks to Jim Frazier, a.k.a. "Mr. Fixit" for keeping the HMMWV running, exterminating wasps, and getting up early lots of mornings just so he could help me lay out traffic cones and then safety driving in both scorching and freezing temperatures. Thanks to Renee Wahl for picking lots of points, moving lots of cones, and enduring the blue goo. Thanks to Kate Fissell, Jim Moody, and Bill Ross for keeping the machines up on and off the vehicle. And thanks to the guys at Gateway Supply, for letting me use their bathroom on those long days spent out at the slag heap.

Thanks very much to those insane people who volunteered to read some or all of my thesis just to help out: Maria Ebling, Marie Elm, Michael English, Terry Fong, and Chris Okasaki. Thanks to my officemates of past and present: Anurag Acharya, Justin Boyan, Fabio Cozman, Keith Gremban, John Hancock, Lily Mummert, Carol Novak, Conrad Poelman, David Pugh, Carlo Tomasi, and Todd Williamson. They've helped me with obscure shellscripts, provided complimentary rubber slugs and beetles to decorate my desk, put up with my tendency to take over more bookshelves than I should, and been wonderful friends.

So many more people deserve my thanks for their help, encouragement, support, and friendship: Omead Amidi, Shumeet Baluja, Harry Bovik, Mei Chen, Stewart Clamen, Mary Jo Dowling, Maria Ebling, Mike & Win English, Emily Kingsbury, Jim Kocher, Mark Maimone, Chris Okasaki, Fred Solomon, Mark Stehlik, Jim Stichnoth, Dafna Talmor, Bennet Yee, Amy Moormann Zaremski, and more.

Finally, thanks to my sister Abby Kay, my parents Gordon and Joyce Kay, and my husband Redmond English, for believing in me, helping me get through the low points, rejoicing with me in the high points, and always being there.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1 Introduction

Supervised TeleRobotics using Incremental Polyhedral Earth reprojection, or STRIPE, is a teleoperation system for a robot vehicle. Using STRIPE, a human operator can accurately control a remote vehicle across very low bandwidth communication links, and communication links with large delays. STRIPE requires no advance knowledge of the terrain to be traversed, and can be used by novice operators with only minimal training. This thesis covers both the creation and development of the STRIPE system from the robotic side, as well as a study of human operators using the system under different conditions and with various interfaces.

*The Need For A Low-Bandwidth High-Delay Teleoperation System*

Vehicle teleoperation systems are generally intended to be used in environments which are unpleasant or even physically dangerous to humans. While it might be appropriate to connect the vehicle to the operator control station with a tether in certain circumstances, tethers can severely limit vehicle range, and often need to be extended and collected with great care to avoid damage.

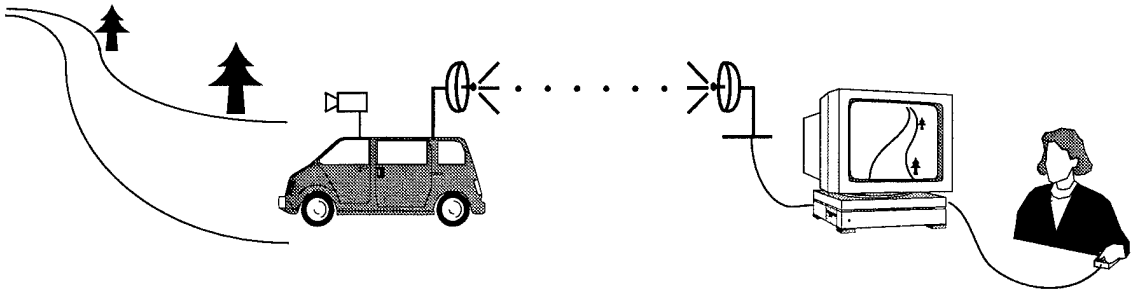
When a teleoperation system is untethered, the bandwidth between the operator control station and the vehicle becomes an issue. A standard grayscale image is about a quarter of a megabyte of data. Reasonable compression schemes may reduce this by a factor of ten or twenty, but even

twelve kilobytes of data is huge when compared to the sixteen to twenty-four Kbits/second of data that standard military tactical links can provide [8]. In addition, experiments with cellular modems in Pittsburgh could not reliably provide more than a 9.6 Kbit/second link. Even when high-bandwidth links are technically possible, they may not be desirable. For example, the military is interested in low-bandwidth teleoperation systems because they reduce the possibility of detection by the enemy.

Delays in communication can be caused by inherent system latency due to such tasks as buffering, command processing, and image compression. Large distances also can contribute to communication delays: the round trip transmission time between the Earth and Mars can be as much as 41 minutes [9].

*How STRIPE Works*

In STRIPE, the operator is presented with a single image taken from the remote vehicle, and is asked to designate a series of points in that image, known as "waypoints," that indicate where the vehicle should go (Figures 1.1 and 1.2). Once the operator is happy with the points selected, the



Images are digitized on the vehicle and transmitted to the operator workstation.

Once the image arrives at the operator workstation, the operator uses a mouse to pick waypoints for the vehicle to track.

**Figure 1.1 The Basic STRIPE System**

points are sent to the remote vehicle, the vehicle begins travelling along the designated path, and the process is repeated. This is very different from traditional teleoperation methods, in which the

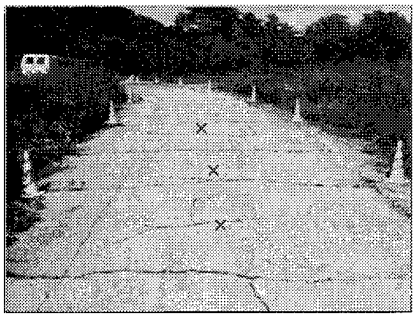


**Figure 1.2 A sample image with three points chosen in it.**

operator views a continuous stream of images and directly controls the vehicle's steering using a steering wheel or a joystick. STRIPE can be used in situations where the communication link has a wide enough bandwidth and low enough delay that the traditional methods of teleoperation are also possible. However, STRIPE's strength is its ability to accurately control a remote vehicle when the communication link makes other forms of teleoperation impossible. STRIPE has been successfully used to remotely control the Carnegie Mellon Navlab 2 (Figure 1.3) with a delay of

**Figure 1.3 The Navlab 2, a converted army HMMWV**

18 seconds between the time the operator finished picking points in one image and the time the next image appeared on the operator's monitor. The time of 18 seconds was chosen to limit boredom on the part of the operator; the distance the vehicle can cover is dependent only on the num-

ber of images transmitted and the locations of the points chosen in those images. The vehicle would travel the same distance if the same points in the image were chosen, even if there were a delay of an hour between images.

In order to convert the two-dimensional image waypoints into steering directions, the points must first be converted into three-dimensional real-world points. STRIPE's polyhedral-earth model allows the position of the real-world points to be automatically updated and corrected while the vehicle is driving. As the vehicle moves, it continuously senses its current orientation, and uses this information to compute the next steering adjustment.

*History*

STRIPE is a part of the Carnegie Mellon University Navlab Project [15][40]. The project began in 1984 with the Terragator, a six wheeled cart. In 1986 the Navlab 1 was built into a commercial van, which was converted to allow computer actuation of the steering as well as the brake and accelerator pedals. In 1991 work began with the Navlab 2, an army Hmmwv that was converted for off-road and higher-speed autonomous driving.

Most of the work on the Navlab vehicles to date had been in the area of fully autonomous control. STRIPE extends the use of the Navlab 2 vehicle to include "semi-autonomous" control, in which the operator provides the high-level decisions about where the vehicle should go, and the vehicle takes over the task of actually controlling the vehicle's steering.

With the introduction of semi-autonomous control to the project came the introduction of a new variable that had never before been considered: the human operator, whose actions often seem at worst, random and at best nondeterministic. It was impossible to study just the robotic side of the STRIPE system and have any real way to evaluate its performance without testing the way real people make use of the system. A series of user studies was developed to test various STRIPE interfaces, and to determine how well novice operators performed different teleoperation tasks.

*Thesis Overview*

STRIPE is technologically simple and inexpensive to implement, and is tolerant to small errors in calibration. Using static images from a single camera, STRIPE accurately follows the

operator's designated path, even on hilly terrain. STRIPE demonstrates the strength of the semi-autonomous approach, allowing a human operator to direct the vehicle, while the vehicle itself monitors the terrain in real-time and makes corrections to the estimated path as necessary. The user-study portion of the work demonstrates that the system is simple enough to use that operators with no previous teleoperation experience can quickly and easily learn to use the system. The user studies also provide a better understanding of the way the system works, and indicate how the robotic side of the problem could be improved to make it more natural for humans to use.

Chapter 2 provides a review of the work in teleoperation to date, including two other systems that take a semi-autonomous approach to the problem. Chapters 3 and 4 detail the robot side of the STRIPE problem, including an analysis of the potential errors involved in this approach. Chapter 5 begins the transition to the actual study of the system, and details some changes that were made to the system following the initial implementation and testing. Chapter 6 details the STRIPE user experiments, and the recommendations for improvements to the system based on the results of these experiments.

# Chapter 2 Previous Work

## 2.1 Classifying Teleoperation Strategies

The field of vehicle teleoperation naturally divides into three classes based on three variables:[1] image update rate, transmission delay, and method of vehicle control. The image update rate is the minimum time between image frames being displayed on the operator workstation. The transmission delay is the amount of time it takes the first bit of data to be transmitted from the vehicle to the operator workstation.[2] This delay may include the time it takes to compress and uncompress image data, encode and decode the transmission, as well as system overhead. Vehicle control can be either direct, in which the operator is directly responsible for commanding the orientation of the vehicle's steering, or semi-autonomous, in which the operator designates the path that the vehicle should follow, and it is up to the vehicle to determine how to adjust the steering appropriately.

1. Note that all of the teleoperation methods that I shall discuss are applicable when the vehicle is not directly visible to the operator ("inside-out control"[26]). Work has also been done in the field of teleoperation of vehicles where no image data is transmitted from the vehicle to the operator, and the operator steers based on their own direct view of the vehicle's movement ("outside-in control"). Outside-in control is very different from inside-out control, and is beyond the scope of this thesis.

2. Strictly speaking it is possible that the transmission delay from the operator workstation to the vehicle could be larger than the delay from the vehicle to the operator workstation, but this is not typically the case.

Many tethered vehicle teleoperation systems fall into the *Continuous and Delay Free* (CDF) category[3], where bandwidth is plentiful, delay is minimal, and operators are in direct control of the vehicle's steering. Several systems are included in the *Nearly-Continuous or Very-Low-Delay* (NCVLD) classification. In this category, operators continue to directly control the vehicle. Although the bandwidth is relatively high and the delay is relatively low, either the bandwidth or delay or the combination of the two is sufficiently poor to cause brief, but noticeable, interruptions in image transmission. Finally, those systems that are *Discrete and Delayed* (DD) have such limited bandwidth and/or such high delay that there is a substantial pause of several seconds, minutes, or more between images. DD systems use a semi-autonomous approach (indirect control). STRIPE is one example of a DD system. Table 2.1 summarizes the way vehicle teleoperation systems are classified in my taxonomy.

| | Control Method | Image Update Rate / Transmission Delay |
|---|---|---|
| **CDF** | Direct | At least 20 frames per second<br>and<br>delay less than or equal to 100 ms. |
| **NCVLD** | Direct | Less than 20 frames per second<br>or<br>delay greater than 100 ms |
| **DD** | Indirect | Any update rate, any delay (typically very low frame rate, and very high delay) |

**Table 2.1 Classifying Vehicle Teleoperation Systems**

## 2.2 Continuous and Delay Free

All Continuous and Delay Free (CDF) systems rely on an operator who directly steers the vehicle. The orientation of the wheels of the remote vehicle is directly controlled by a steering wheel or joystick at the operator workstation.

3. The categories "Continuous and Delay Free," "Nearly-Continuous or Very-Low-Delay," and "Discrete and Delayed" are my own classifications developed for use in this thesis.

In most CDF systems, the operator workstation has one or more monitors that display images transmitted from the remote vehicle, along with controls similar to the steering and speed controls in a car. The operator has the impression that he or she directly controls the steering and speed of the remote vehicle. To make the control area as realistic as possible and to eliminate certain variables in testing, some systems even have a control area that can be fitted on the vehicle for comparing remote and on-board driving.

CDF systems are characterized by a continuous stream of images (at least 20 frames per second) with virtually no transmission delays (delays are at most 100 ms). The specific bandwidth requirements for a given CDF system vary depending on system details (number of displays, image resolution, etc.), but is always high enough to present at least 20 frames per second to the operator.

The 20 frames per second restriction was chosen because humans need to see about 20 frames per second for the illusion of continuous motion [5]. As the frequency of images drops below 20 frames a second, individuals initially notice a flicker. As the frequency continues to drop, they become aware that they are viewing a sequence of distinct images.

Image data takes up a great deal of space. A typical 512 x 480 8-bit grayscale image is about a quarter of a megabyte of data. Image compression techniques can be used to reduce the size of the data somewhat: algorithms that can perfectly reconstruct the original image average about a 50% reduction in data for typical images [1]. Images can be further compressed (with loss of some data) down to between 2% and 10% of their original size without affecting the apparent quality as viewed by the human eye [41]. Compressing a typical image down to 2% of its original size still produces about a 5 KB image, a significant amount of data.

CDF systems transmit surprisingly large amounts of image data. Most CDF systems run at 30 frames per second, the standard rate for television transmissions. Thus CDF systems need enough bandwidth to transmit about 7.5 MBytes of data per second, for uncompressed grayscale images. Although image compression and reduced resolution images can reduce this bandwidth somewhat, the resulting data is still well above the 10 KBytes per second that can be transmitted along analog phone lines using the best modems available today.

CDF systems also require a latency of less than 100 ms. With a latency of less than 100 ms, the system provides the human operator with the perception of causality [5]. In other words, the operator believes that there is no delay between the turning of the steering wheel or movement of the joystick at the operator control station and the time at which that steering command takes effect at the remote vehicle.

It is useful to study a few examples of CDF systems to provide some insight into the types of difficulties that CDF operators face.

### 2.2.1 Naval Ocean System Center

In an attempt to improve the remote operator's performance to approximate that of a normal driver, some systems try to give the operator the feeling of actually being on the remote vehicle. The system developed at the Naval Ocean System Center (NOSC) attempted to impart this feeling of "telepresence" to the operator [2][16][24][33] through a special helmet that provided both stereo image data and stereo audio data. A corresponding anthropomorphic head on the vehicle was used to gather the visual and audio data. The motion of this robot head followed the movement of the operator's head, so that operators perceived the same things they would have perceived had they actually been on the vehicle.

The vehicle itself was a converted army HMMWV, connected to the control station by a fiber optic tether that allowed for remote communications at a distance of up to 30 km. The tether was very high bandwidth, allowing simultaneous transmission of up to 200Mb/s from the vehicle to the operator control station and 100Mb/s from the operator control station back to the vehicle. Although the explicit frame rate and latency are not given, it is reasonable to assume from the description of the system and the tether that it falls into this category.

#### 2.2.1.1 Anecdotal Results

*Operators Get Lost*

Although one would expect such a high degree of telepresence to give the operators a good understanding of their environment, tests have shown that this is not the case. Operators of the NOSC dune buggy had difficulty judging the location and orientation of the vehicle relative to

known landmarks in the real world. In other words, just like the Sandia operators, NOSC operators tend to get lost.

#### 2.2.1.2 Empirical Experiments

*Local vs. Remote Driving*

As would be expected, operators consistently performed worse when remotely operating the NOSC vehicle than when they sat in the vehicle itself and drove normally. In all cases, remote driving times were at least twice as long as the times for the operator in the vehicle.

*Direct Driving with Limited Field of View*

It was hypothesized that the poor performance of remote drivers was due to the limited horizontal field of view (60 degrees) that the head-mounted display provided, since the normal human horizontal field of view is 204 degrees [20]. However, on-board drivers wearing a helmet with a face shield permitting only a 60 degree horizontal field of view performed just as well as they did when their field of view was unimpaired.

*Direct Driving with Limited Image Resolution*

Degraded image resolution presented to the remote driver was also hypothesized as a cause of poor performance. Camera resolution was measured to be approximately equivalent to 20/80 human vision. On-board drivers were tested with special goggles that degraded their vision to be 20/40. No difference was discovered between their normal performance and their performance with the goggles. The goggles were then further degraded to between 20/50 and 20/100. Two out of the three drivers were still able to match their normal performance with the goggles, although the third driver was not able to complete the course without becoming disoriented.

### 2.2.2 Army Research Laboratory

Work at the Army Research Laboratory (ARL) in teleoperation has studied the effects of varying fields of view on the operator of a remote vehicle system [12]. Using cameras with three different fields of view: 29 degrees, 55 degrees, and 94 degrees, researchers compared the ability of remote operators with that of in-vehicle operators. Again, the image frequency and latency of the system are not reported, but it appears to be a DD system. In all but the obstacle avoidance seg-

ment of the study, on-board operators performed statistically significantly better than remote operators. They found that the 55 degree field of view lens provided the best compromise among field of view, image distortion, and image resolution. It is interesting to note that while most of the participants in the study reported being uncomfortable with the distorted image the 94 degree field of view camera provided, they preferred it during the obstacle avoidance segment of the tests.

ARL has also investigated the effects of image resolution and number of grey levels on teleoperation; however the results of this work have not been published[4].

### 2.2.3 Discussion

The most striking point about CDF systems is how poorly operators perform the task of remote driving. Even in the case of the NOSC system, where vast quantities of data are available to operators, their remote driving performance is significantly worse than their on-board performance with apparently equivalent visual impairments. Of special concern is the fact that operators in the NOSC study lost their way, a severe handicap.

## 2.3 Nearly-Continuous or Very-Low-Delay

"Nearly-Continuous or Very-Low-Delay" (NCVLD) systems require the operator to be in direct control of the vehicle's steering. Direct systems that present the operator with less than 20 frames per second, or have a latency of at least 100 ms are considered to be NCVLD. Systems that are directly controlled by the operator most of the time, but that allow the vehicle to veto or adjust the operator's steering command to avoid obstacles also fall into the NCVLD category. There is no explicit upper limit on the latency or lower limit on the frame-rate. However it is clear that operators need to receive images with a frequency that is measured in seconds rather than minutes or hours. The important point about the limits is that humans are still able, with some degree of success, to directly control a remote vehicle on a NCVLD system, despite their consciously recognizing that the images that they see are significantly delayed and/or discrete.

4. Schipani, Salvatore, Personal Conversation, US Army Research Laboratory, Human Research and Engineering Directorate, Aberdeen Proving Ground, MD, October 1993.

### 2.3.1 Sandia National Laboratories

Sandia National Laboratories' scientists have had extensive experience in the field of vehicle teleoperation [3][25][26][27][28][29][30][31]. They have a large fleet of vehicles, ranging from small vehicles for indoor teleoperation, to very large commercial vehicles for both on-road and off-road testing. They have experimented with many different camera configurations: single and multiple camera systems, fixed and steerable cameras, color and monochrome. They were able to provide 3 Mbits/sec bandwidth and a data rate of 30 frames/sec, although they have done some experimentation into lower bandwidths and frame rates[26]. For their Jeep system, they reported a mean lag of 235 msec between the time a steering command was issued and the time the remote vehicle responded [28].

#### 2.3.1.1 Anecdotal Results

*Operators Lose Their Way*

One notable difficulty that operators experienced was the tendency to get lost while remotely operating the vehicle. Operators would frequently lose their way, and not be able to report their location with respect to major landmarks, map features, or compass directions. Operators were also unable to plan a path back to their starting position without assistance, even when they were provided with a map of the course.[26]

*Operators Oversteer*

Remote operators initially tend to oversteer when they do not instantly see a response to their control commands in the images coming from the vehicle. Typically, the operator turns the steering wheel slightly, does not immediately see the expected change in the vehicle, and so increases the steering wheel input until the vehicle begins to respond. At this point, the vehicle begins to turn more than the operator intended, and so the operator attempts to compensate by steering in the other direction and the pattern repeats itself, with the vehicle oscillating left and right along the desired path. However, operators become accustomed to this effect in a matter of minutes and their performance improves significantly. [26]

*Image Resolution*

McGovern [26] reported that operators were able to accurately control remote vehicles when provided with low resolution images, although the actual resolution was not presented. He did note that high resolution images appear to be necessary for off-road teleoperation, and teleoperation in an area with obstacles of varying size and type.

*Difficulties In Understanding the Remote Scene on a Monitor*

Remote operators tended to misjudge the data that they received on their monitors. They tended to misjudge distances, believing they were further from objects than they actually were, and missed small negative obstacles (e.g. holes, ditches, etc.). They also tended to misjudge the magnitude of tilt and roll of the vehicle.[26]

*Field of View*

Operators had difficulty driving in a restricted space with the single 40 degree field of view camera. They found it much easier to turn corners when two additional cameras were installed to provide a 120 degree field of view presented on three monitors. [26]

#### 2.3.1.2 Empirical Experiments

*Accidents*

Sandia has the distinction of being one of the few institutions to report accidents in their literature. This is probably due to the robustness of their system more than to anything else; most test vehicles have fragile and expensive equipment on board that would be easily damaged by minor collisions. One of the vehicles at Sandia was an all-terrain vehicle that was robust enough to survive multiple accidents.

All of the accidents were rollovers, and all but one of these accidents occurred on an off-road course with steep slopes and high-banked corners. Most of the rollovers occurred when the vehicle was travelling at its maximum speed of 10 to 15 m.p.h. Typically, the operators of the vehicle said that they had not realized the magnitude of the tilt and roll of the vehicle immediately before the accident and were surprised by the roll-over. [26][27]

*Automatic Camera Pan*

Experiments were performed in which the camera pan was automatically adjusted to match the steering angle of the vehicles tires. This provided no statistically significant improvement in obstacle recognition and obstacle detection, although it did aid operators negotiating tight corners and avoiding obstacles.[26]

*Use of Color Images*

Use of a color camera was shown to be a statistically significant factor in earlier detection of obstacles. Obstacles were detected an average of 9 feet earlier with the color displays than with monochrome; obstacle size was the major factor in early detection. Large obstacles were detected on average 22 feet earlier when using color, although small obstacles were detected on average less than 5 feet earlier with color. [30]

Color was also shown to be a statistically significant factor in an operator's ability to judge the distance to a pair of objects, as well as the distance between the objects themselves, in order to determine whether the vehicle could be maneuvered between them.[28]

### 2.3.2 Oak Ridge National Laboratory

Anecdotal evidence suggests that driver performance in NCVLD systems declines as the length of the time necessary to transmit an image grows. However, some successful systems exist. For example, work at the Oak Ridge National Laboratory on improving compression techniques has reduced the delay between images sufficiently to enable direct control of vehicle steering over narrow bandwidth links (about 64 Kbps) [8]. Drivers were able to drive on a bumpy dirt track at speeds of up to 15 m.p.h., even though no more than five images (compressed at an average of 125:1) were displayed each second and each had a transmission delay of about one second.

### 2.3.3 Carnegie Mellon University

Krotkov et al. [21] have developed the concept of *Safeguarded Teleoperation* for a lunar rover, in which there would be a communications latency of about 2.5 seconds between the vehicle and operator. Safeguarded teleoperation provides the remote operator with direct control of the vehicle most of the time, enabling the operator to indicate both direction of travel and vehicle

speed. In order to prevent the vehicle from driving into obstacles, because of operator miscalculations due to the delay or malicious intent, the vehicle maintains the power to override the operators commands, using on-board sensors to determine when the vehicle is approaching a dangerous situation. Experiments using a human "wizard" to provide the system override, and a maximum speed of less than half a meter per second showed that most of the users were to complete an obstacle course successfully, hitting only one obstacle per trial on average. However, operators reported that the addition of the wizard to the system did not reduce fatigue or frustration with the task.

Initial experiments were also performed in a simulated environment using a *predictive display* in which two lines were projected into the image indicating where the vehicle would probably be 5 seconds in the future, i.e. the time when the operators' current commands would reach the vehicle. These experiments show that users drove faster, oversteered less, and hit fewer obstacles even when no wizard was present.

The ability for a vehicle to override operator commands in order to prevent collisions is valuable for all teleoperation systems that have a significant delay in image transmission. However, without a terrain map of the area to be explored, the lines drawn by the predictive display will not be accurate on non-planar terrain, for the same reasons that the FELICS system has difficulties. Even on planar terrain, the system will make erroneous predictions if the vehicle's wheels skid during a maneuver.

Direct control of a vehicle using safeguarded teleoperation may be feasible on the Moon, but it is unlikely that anyone could use a direct teleoperation system to control a vehicle on Mars, where the transmission latency is measured in minutes rather than seconds, without reducing the velocity by several orders of magnitude.

### 2.3.4 Massachusetts Institute of Technology

Verplank [39] studied the effect of a predictive display[5] in the teleoperation of a simulated underwater vehicle moving at a fixed height above the sea floor with a one second transmission

5. Sheridan & Verplank referred to these as "predictor displays".

delay and a maximum frame rate of 8 seconds per image. Although this is different from the general problem that is considered in this thesis, one result of the study is worth noting.

Operators were tested using two modes. In the first, new images were transmitted every 8 seconds. In the second, a new image was not transmitted until requested by the operator. Surprisingly, operators were less confused and required fewer images using the second mode, with or without the predictor display. Operators using this mode adopted a "move and wait" strategy, in which they would maneuver the vehicle a certain distance, and then request a new image and wait to see it before proceeding.

### 2.3.5 Lunokhod

No discussion of NCVLD teleoperation would be complete without covering the Soviet Lunokhod vehicles, the only extra-terrestrial vehicles actually teleoperated from the Earth [6][11][18]. Lunokhod 1 and 2 were 8 wheel, skid-steered vehicles that were teleoperated on the moon in the early 1970's. The Lunokhods were operated by a 5 man team: commander, driver, navigator, systems engineer, and radio operator. Each team member sent signals to the vehicle. The driver had the options of both forward and reverse, and continuous as well as stop and start motion. The communications delay made controlling the vehicle extremely difficult, and it was said that many highly experienced drivers were unable to perform the task.

With the exception of the lunar night, when the vehicle was stationary, Lunakhod 1 achieved an average velocity of 4.8 m/hr. Lunakhod 2 achieved an average velocity of 27.9 m/hr, although the vehicle could achieve a cruising speed of 7 km/hr. It seems reasonable to surmise that at least part of the reason for such a low average velocity was the difficulty of the driving task.

### 2.3.6 Discussion

The techniques used in CDF and NCVLD teleoperation are essentially the same. The only difference is that in NCVLD teleoperation, the operator is required to teleoperate the vehicle without the perception of immediate causality. The problem with operators oversteering that was present in the Sandia studies is clearly the result of the system latency.

## 2.4 Discrete and Delayed

Our discussion so far has dealt with fairly-high-bandwidth and very-high-bandwidth systems that have very short transmission delays. We have yet to consider systems for remotely operating a vehicle on Mars, for example, where round trip transmissions from Earth and back vary between 8 and 41 minutes at the speed of light [9], or very low bandwidth transmission links where even the most compressed images would take several seconds to transmit.

Depiero [8] has shown that human operators can directly and effectively control the steering of a remote vehicle if they are provided with approximately five images per second. Even with heavily compressed images, many communication links do not provide enough bandwidth to transmit a single image per second. Non-line-of-sight tactical communication links can transfer about 2-8 KB/sec [8], and cellular modems transfer less than 2 KB/sec. Some Mars mission plans allow as few as tens of *bits* per second [17]. Clearly, a transmission delay of one hour between image capture and display on the operator workstation is much too large to enable any sort of realistic direct steering of a vehicle. Even a transmission delay of 1 minute per image is too long. It seems as though transmission rates of anything over several seconds per image make direct driving impossible, and some kind of automatic vehicle control becomes necessary.

Those systems that are semi-autonomous, i.e. the operator is not in direct control of the vehicle's steering, will be referred to as "Discrete and Delayed" (DD). The STRIPE system falls into this category. In DD systems the operator gives more general instructions to the vehicle, and the vehicle computes the appropriate steering directions. In most of these systems, the general instructions are in the form of a list of 3D points, known as *waypoints* in the vehicle or world coordinate frame. Given a list of 3D waypoints, the vehicle can compute the appropriate steering angles to enable it to head towards its goal. While DD systems could theoretically have a high frame rate and a low delay, their strength is in situations where the frame rate is low and the delay high.

### 2.4.1 Dynamic System Technologies

The Feedback Limited Control System (FELICS) [37][38], developed by Dynamic System Technologies, enables an operator to remotely drive a vehicle with a single image every 3.5 sec-

onds. The system requires a bandwidth of less than 10 kbits/sec with 50:1 image compression. A triangular "puck," a cursor that represents the vehicle's location, is superimposed on the image, initially just in front of the vehicle's position. The operator uses a knob (to control heading) and a stick (to indicate speed) to "drive" the puck through the image, thereby establishing the path the vehicle should follow. At predefined intervals, a marker is placed in the image along the puck's path to mark a waypoint (i.e., a point towards which the vehicle should steer).

Each 2D image waypoint corresponds to an infinite number of 3D real-world points, and so the system must somehow determine the one 3D point in the real world that the 2D point really represents. This conversion is accomplished as follows: At the time of image capture, the location of the vehicle's "groundplane," the plane on which the vehicle's wheels rest, is noted. Figure 2.1 shows a side view of the vehicle, the path to be traversed, and a vastly oversized pinhole camera

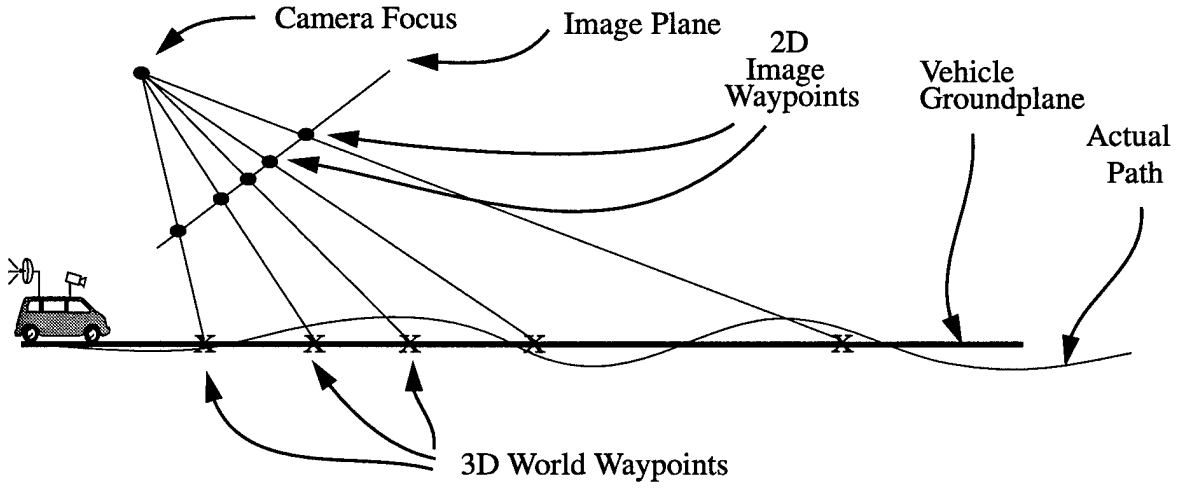



**Figure 2.1 FELICS Flat-Earth Reprojection**

representing the camera on the vehicle. Points on the image plane (i.e., 2D waypoints) can be reprojected through the focus out into the world, and their intersection with the groundplane (represented by the x's) is considered to be their location in the 3D world. The resulting 3D waypoints are transmitted back to the vehicle where they are used to control the steering.

Although FELICS works well enough at 3.5 seconds per image to keep the vehicle on a wide road, its flat-earth projection method causes errors on non-planar terrain. These errors can become increasingly problematic as image frequency decreases. For example, consider the following scenario: Suppose that the vehicle is moving at a constant speed of 1 m/sec. Assume that the system uses a 5 m lookahead distance (i.e., the vehicle always steers towards a point 5 m ahead of itself). Further, assume that it takes 9 sec for the image to be transmitted from the vehicle to the operator workstation, and that 1.5 sec after the image arrives the appropriate waypoints have been selected and are transmitted back to the vehicle, which takes another 0.5 sec. To summarize:

- Speed: 1 m/sec
- Lookahead: 5 m
- Image Transmission time: 9 sec
- Path Picking: 1.5 sec
- Reply Transmission time: 0.5 sec

Given these assumptions, what is the minimum distance ahead of the vehicle that we need to predict the location of the path so that we can move continuously at our desired speed of 1 m/sec? Consider the time line shown in Figure 2.2. At time t=0, transmission of the first image begins from the vehicle (black bar). At time t=9, the image has arrived at the operator workstation and the operator begins to place waypoints using this image. These waypoints are transmitted at t=10.5 and arrive back at the vehicle at t=11. Meanwhile, as soon as transmission of the first image concludes at t=9, transmission of the second image begins (gray bar). As shown in Figure 2.2, the waypoints from the second image arrive at the vehicle at t=20. Thus, between t=9 and t=20, the vehicle is being steered based on an image digitized at t=0, and so we must be able to plan a path up to 20 seconds ahead of the vehicle. In 20 seconds, the vehicle moves 20 m (20 sec x 1m/sec), and the vehicle needs points an additional lookahead distance of 5m. Thus, the minimum distance ahead of the vehicle that we need to predict waypoints is 25m.

Now consider the scene depicted in Figure 2.3 which has been drawn to scale. The initial groundplane of the vehicle is horizontal, but the road ahead has a slight slope of 3 degrees. Suppose a sequence of points has been chosen by the operator, and a point 50 meters ahead of the vehicle on the road has been designated as the point where the vehicle should stop. With

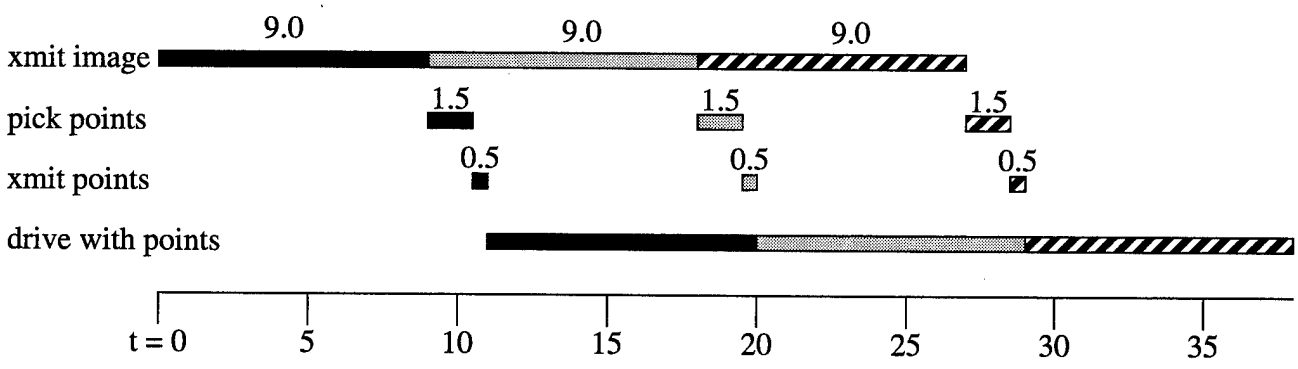



**Figure 2.2 Teleoperation Time Line**

FELICS's simple flat-earth approach, all points are projected onto the current groundplane, and the vehicle will assume that its final waypoint for this image is the stopping point, which it will incorrectly predict as being located 25m ahead of its initial position. (A more detailed explanation of this effect is described in Section 4.7.2.)

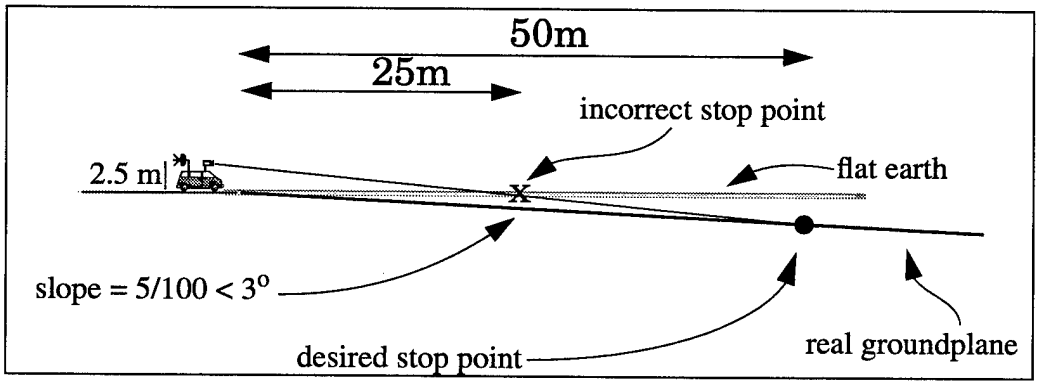



**Figure 2.3 Flat Earth Reprojection.**

## 2.4.2 Jet Propulsion Laboratory

Two types of DD systems have been developed at the Jet Propulsion Laboratory (JPL): Computer Aided Remote Driving, and Semi-Autonomous Navigation [4] [7] [9] [23] [42] [43] [44] [45]. JPL is primarily interested in teleoperation systems that function over links with delays, as in space applications.

#### 2.4.2.1 Computer Aided Remote Driving

Computer Aided Remote Driving (CARD) [4] [7] [9] [23] [42] [43] [44] [45] is a system for semi-autonomous control of a remote vehicle using stereo image data. In CARD, a stereo pair of images is transmitted to the operator from the vehicle. These images are presented to the operator on a stereoscopic display, and the operator uses a 3D cursor to designate *3D real-world* waypoints in the scene. The 3D waypoints are transmitted back to the vehicle, and are used to compute the appropriate steering angles. Because the CARD waypoints are chosen in a 3D image, no 2D to 3D conversion is necessary, and no assumptions must be made about the shape of the terrain. The vehicle is expected to travel about 20 meters for each stereo image pair sent to the operator.

CARD's primary objective is to provide a means for interplanetary vehicle teleoperation. In most interplanetary scenarios, there is both a delay due to distance as well as a limited bandwidth link. The CARD method is effective in situations where transmitting two images has approximately the same cost as transmitting a single image, as in high-bandwidth space applications. However, when the delay is not only due to distance, but also to low-bandwidth transmission, sending two images might take twice as long as sending a single image; this may make the delays unacceptable.

#### 2.4.2.2 Semiautonomous Navigation

In Semiautonomous Navigation (SAN) [9] [23] [42] [43] [44] [45] a topographic map produced from orbital images is used by a human operator to plan a path for the remote vehicle. The vehicle computes a local topographic map using on-board sensors such as stereo cameras or a laser range finder, and determines the correlation between its map and the one used by the operator. The remote vehicle plans a new path, based on the operator's path, but revised to account for small obstacles that may have been missed in the lower resolution orbital images.

The remote vehicle follows its revised path a short distance (on the order of 10 meters), and then does another reconstruction of the local environment and the process repeats. In this way, the vehicle is expected to be able to travel at an average speed of approximately 10 cm/sec.

### 2.4.3 NASA Ames Research Center

NASA Ames is also interested in interplanetary teleoperation, and they have developed a system called the Virtual Environment Vehicle Interface (VEVI) [10][17][34]. VEVI is used for previewing, planning, and describing high-level task descriptions for a mobile robot. In VEVI, a local model of the vehicle's environment is generated using sensors on the vehicle, and this is transmitted from the vehicle back to the operator workstation. The operator views a 3D model of the world on a stereoscopic display, complete with a 3D model of the vehicle in the appropriate location. To plan a path, the operator moves a copy of the 3D vehicle model along the desired 3D trajectory. The operator can also control the field of view and viewpoint for the display. Once the operator has designated a path, that path is transmitted back to the vehicle, and the vehicle uses this 3D data to plan its trajectories.

VEVI is intended to be easy for operators to use and comprehend, but this means that it requires some complex software and hardware to run. It uses 3D rendering on a stereoscopic display, along with significant computation to determine what moves of the vehicle are reasonable. It also requires a fairly detailed model of the vehicle (e.g. about 1000 tri-polygons were necessary to model the Dante II robot), and equipment for sensing 3D data on the vehicle.

### 2.4.4 Discussion

Although a surprising number of DD systems have been developed, they all have various limitations that suggest that a better system is possible. FELICS's inability to perform on all but the flattest of terrain under low bandwidth and high delay conditions makes it inappropriate for most uses. Semi-autonomous navigation requires detailed satellite imagery. SAN and VEVI require that 3D data is acquired from the remote vehicle. What is needed is a system that can accurately drive with a single camera and yet has a simple operator control station. The STRIPE system, which was developed for this thesis, is such a system.

# Chapter 3 The STRIPE System

## 3.1 Requirements For A New Discrete and Delayed Teleoperation System

None of the discrete and delayed teleoperation systems discussed in the previous chapter solves all of the difficulties associated with this task. The STRIPE teleoperation system is designed to provide a better system.

The low-bandwidth requirement suggests that the system should minimize the amount of data to be transferred between the vehicle and the operator workstation. Thus, monocular greyscale image data is preferred. If time were no object, a simple flat-earth reprojection system similar to FELICS would do a fine job. Consider the algorithm described in Figure 3.1. This is essentially the FELICS system with a very limited amount of movement per frame. The 5 cm distance is fairly arbitrary, but the idea behind it is not. Flat-earth reprojection performs poorly if the world is not flat. Points very close to the vehicle are predicted with reasonable accuracy, but as the terrain varies and the locations of points move further away the error can become overwhelming (see Section 4.7.2).

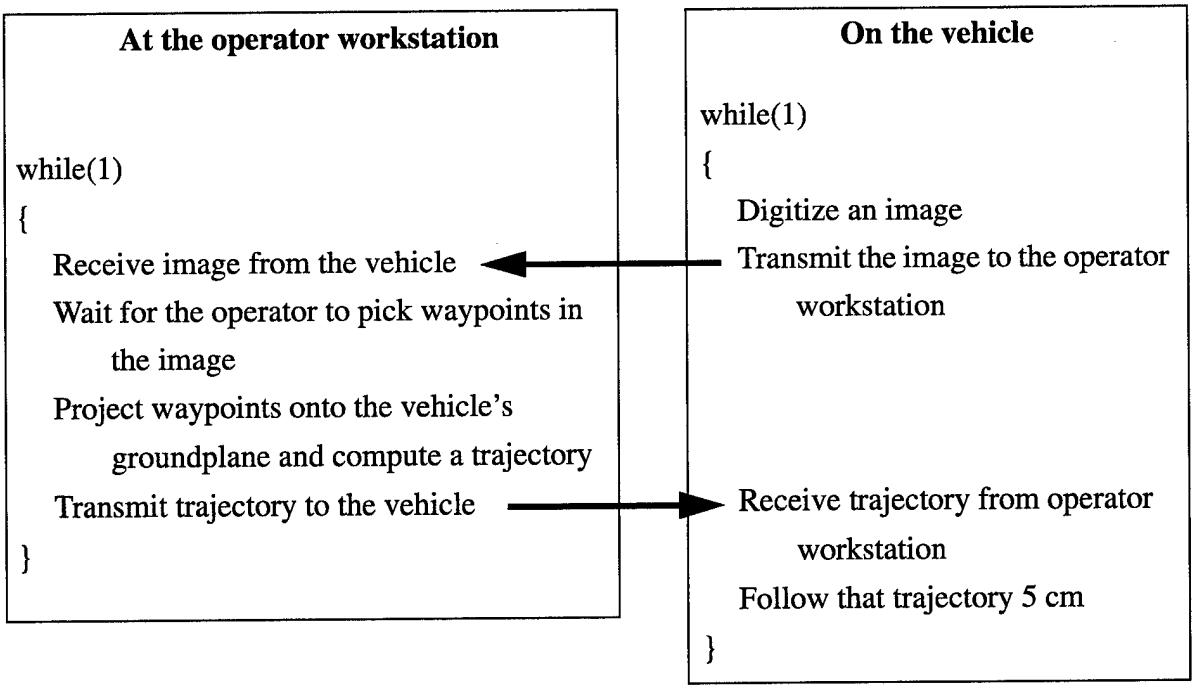



**Figure 3.1 A Simple Discrete and Delayed Teleoperation Algorithm**

In most applications, however, time does matter. If the system is to make any sort of reasonable progress under very-low-bandwidth and very-high-delay conditions, the vehicle must be able to accurately travel a reasonable distance on a single command. And although it is easy to debate the definition of reasonable, it seems that the order of magnitude should be meters, rather than centimeters.

## 3.2 The Basic STRIPE System

STRIPE is a system for vehicle teleoperation under discrete and delayed conditions that *does* enable the vehicle to accurately traverse several meters on a single command.

The basic STRIPE system consists of three modules:

- The "Image Capture Module," which runs on the vehicle.
- The "Operator Workstation Module," which runs at the base station.
- The "Main STRIPE Module," which runs on the vehicle.

A single image from a camera mounted on the vehicle is digitized by the Image Capture module. The image is then transmitted from the vehicle to the Operator Workstation module, where it is displayed on a monitor. The operator then uses a mouse to pick a series of image *waypoints* that designate the path the vehicle should follow (see Figure 3.2). These 2D points are transmitted back to the Main STRIPE Module on the vehicle, and the Image Capture Module is notified that it should begin to send a new image to the operator.

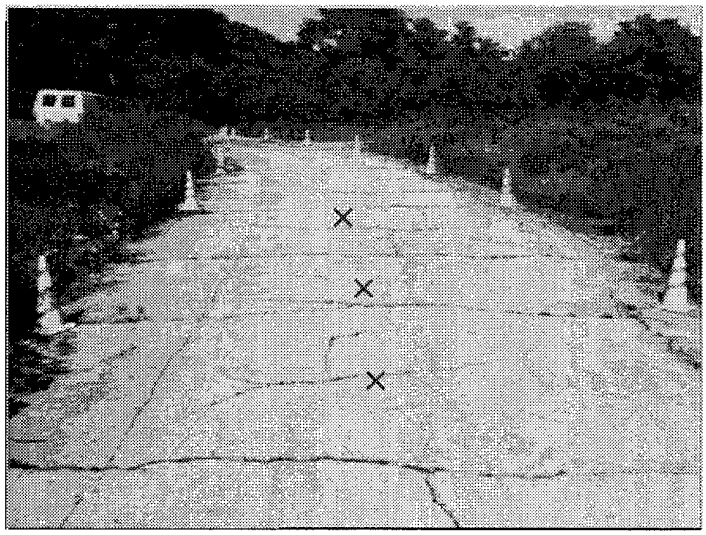


**Figure 3.2 Points chosen in an image.**

In order to compute the vehicle's steering, the Main STRIPE Module must convert these 2D image waypoints into 3D real-world points. Clearly, a simple flat-earth reprojection, such as that used by FELICS, is not sufficiently accurate over these longer distances.

When the 2D waypoints are transmitted to the vehicle, they are initially projected onto the vehicle's current groundplane, using a pinhole camera model. The resulting 3D waypoints are used to initiate steering of the vehicle, and the vehicle starts to move. Several times a second the vehicle re-estimates the location of its current groundplane by measuring vehicle position and orientation. The original image waypoints are then projected onto the new groundplane to produce new 3D waypoints and the steering direction is adjusted appropriately. This reproject-and-drive procedure is repeated until the last waypoint is reached, or until new waypoints are received. An outline of this algorithm is shown in Figure 3.3.

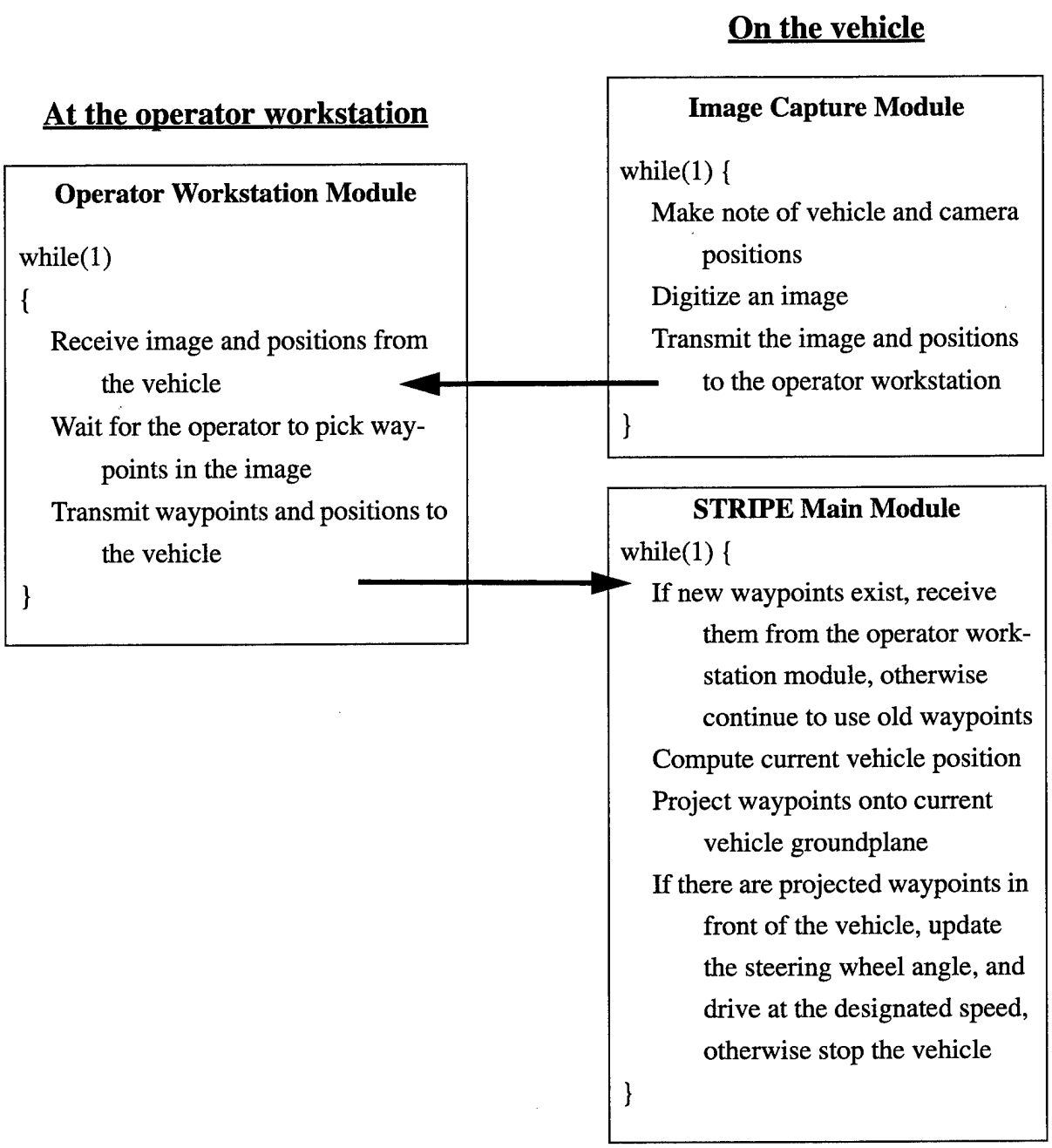



**Figure 3.3 The Basic STRIPE Algorithm**

Consider the example scenario shown in Figure 3.4. The oversized pinhole camera represents the camera on top of the vehicle, with the 2D waypoints on the image plane. The intersection of the projections of each of the 2D waypoints with the actual terrain is the location of the "actual" 3D waypoint the operator intended the vehicle to track

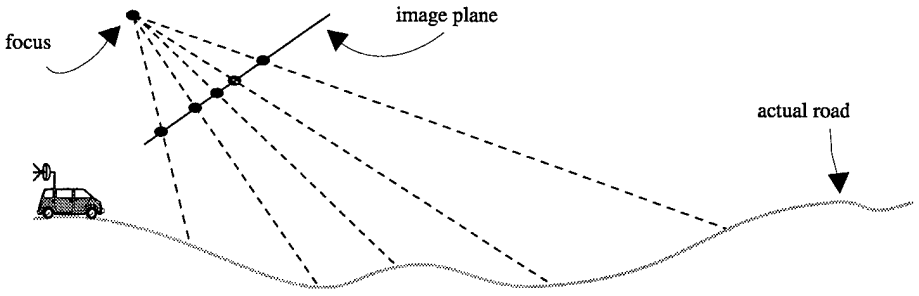



**Figure 3.4 Side view of a road and the corresponding image, with designated waypoints.**

Initially, the STRIPE Main Module projects the 2D waypoints onto the vehicle groundplane, as shown in Figure 3.5. After the vehicle has begun to move, it updates its prediction of the waypoints by reprojecting the original waypoints onto the new groundplane as shown in Figure 3.6. Figure 3.7 shows the vehicle after it has passed several of the waypoints.

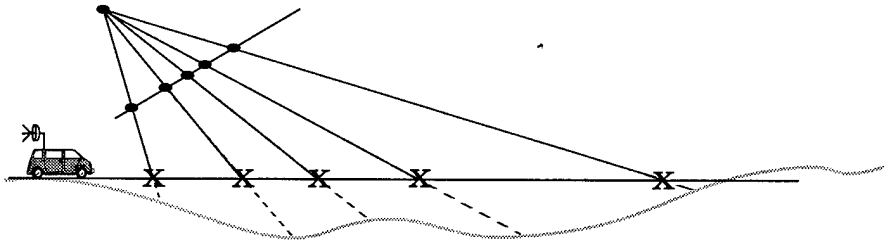


**Figure 3.5 The 2D waypoints are initially projected onto the vehicle's groundplane. The x's represent the location of the projected point.**

STRIPE has no advance knowledge of the 3D locations of all of the waypoints. However, as the vehicle approaches a particular waypoint, the vehicle's groundplane becomes an increasingly accurate approximation for the plane that the waypoint lies on. By the time the vehicle needs to steer based on that particular waypoint, it has a fairly accurate approximation of where that point lies in the 3D world (see Section 4.7.2). Note that all of this movement was based on a single image (and so the pinhole camera does not move throughout this segment).

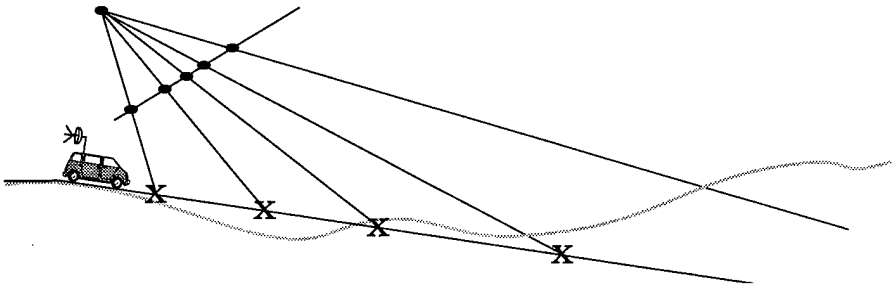


**Figure 3.6 The same waypoints are reprojected onto the vehicle's new groundplane.**

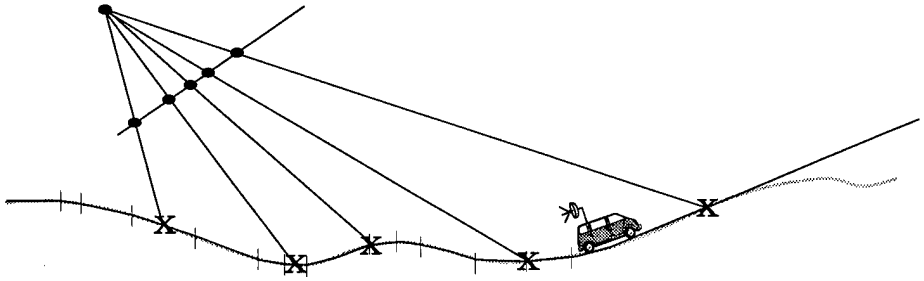


**Figure 3.7 The vehicle continues its reproject-and-drive procedure. The path behind the vehicle has been divided into small planar regions.**

To make the path smoother, once the 2D waypoints arrive at the vehicle, a cubic spline through the points is computed using code from Numerical Recipes in C [36], and this spline is evaluated at 100 different points to generate a smooth path. If the operator only picks a single point in the image, an additional point at the bottom center of the image is transmitted as the initial point.

While the STRIPE Main Module is driving using the 2D waypoints that have been sent from the operator, the Image Capture Module digitizes a new image and sends it to the Operator Workstation Module. If the link between the Operator Workstation Module and the vehicle is very low bandwidth or has a high latency, it may take some time for this image to arrive at the Operator Workstation. While the image is being transmitted, and then while the operator is picking points in that image, the vehicle continues to follow the path designated in the previous image. If the

vehicle runs out of points in this path, it stops. However, if the link between the vehicle and Operator Workstation is sufficiently high bandwidth and low latency, the vehicle never runs out of points before the next set of points is sent, and so the vehicle can move continuously.

Because the vehicle may continue to move after an image has been sent, it may have moved past some or all of the 2D waypoints that the operator picked in that image. In this case, any points that are behind the vehicle are assumed to have been followed already and are ignored.

When the Image Capture Module is signalled to send a new image to the Operator Workstation, it checks the vehicle's position and compares that with the vehicle's position the last time an image was sent. If the vehicle has not moved, and the camera has not been panned or tilted since the last image was taken, the Image Capture Module waits for up to two seconds before digitizing an image. (Two seconds was a reasonable compromise between giving the operator a new image as quickly as possible, and giving the operator an image that might be more useful.)

This situation always occurs at the beginning of a STRIPE run. The vehicle is initially stationary and the first image is sent to the Operator Workstation. The operator's 2D waypoints are sent to the STRIPE Main Module, and the Image Capture Module is informed that it is time to digitize a new image. If an image were to be digitized immediately, the vehicle would have moved almost no distance at all and the next image would look identical to the first. So the Image Capture Module waits a couple of seconds to allow the vehicle to start following any new path that it might have just received, and then digitizes an image that will show a little bit of progress along the path of 2D waypoints that was just received.

It is important to emphasize that the incremental polyhedral-earth assumption that STRIPE makes is very different from standard flat-earth systems. Because the STRIPE Main Module is directly in charge of steering the vehicle, it does not need to plan all of the steering commands it will use in advance. Instead, it can compute the steering direction as it needs it. In the STRIPE path tracking algorithm, a "look-ahead distance" is a predefined constant value.[1] The vehicle

1. The lookahead for the tests conducted in this thesis was 3.9 meters, a point just in front of the Navlab vehicle.

steers by finding the point on the desired path that is approximately one look-ahead distance away from the vehicle, and heading towards that point.

STRIPE's continuous recomputation of the groundplane gives it a distinct advantage over systems like FELICS which assume a flat-earth for an extended period of time. Let us review the example of Section 2.4.1 and see how the STRIPE continuous reprojection performs.

- Speed: 1m/sec
- Lookahead: 5 m
- Image Transmission time: 9 sec
- Path Picking: 1.5 sec
- Reply Transmission time: 0.5 sec

Recall that under the above conditions the vehicle needs to predict waypoints 25m ahead of the vehicle location at the time of image capture. The FELICS system incorrectly places the vehicle stop point 25 m short of the actual point.

The STRIPE vehicle *eventually* needs to predict points 25 m ahead of the vehicle location at the time of image capture. But at any given time, it only needs to predict points that are one lookahead distance in front of the vehicle, i.e., 5 m. Figure 3.8 is the STRIPE scenario corresponding to that of Figure 2.3, and is also drawn to scale. Here the STRIPE vehicle is about to switch onto a new groundplane, that slopes down at three degrees. A point predicted one lookahead distance ahead will currently be 0.45 m closer than it should be. But as soon as the vehicle moves on to the new groundplane, subsequent points on that groundplane will be correctly predicted.

This example is particularly illustrative of the problems that even a very minor change in groundplane orientation can cause. By restricting the flat earth assumption to the look-ahead distance, the accuracy can be significantly improved.

In the version of STRIPE used for the operator testing, the vehicle lookahead was set to a point almost immediately in front of the vehicle. This short lookahead was chosen in order to keep the vehicle as close as possible to the path designated by the operator. (The longer the looka-

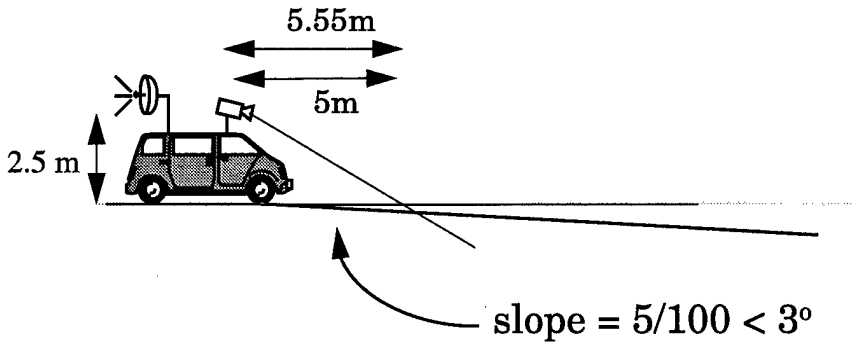



**Figure 3.8 STRIPE reprojection on uneven terrain**

head distance, the more the vehicle would cut corners, something that is undesirable in this sort of system.)

The origin of the Navlab 2 vehicle is on the ground at the center of the back axle. It was thought that operators would find picking points for the back of the vehicle unnatural, and so the vehicle actually stopped when the front of the vehicle had just crossed the last waypoint. This was accomplished by throwing out the last three meters of waypoints at exactly the time that there were only three meters left (because, of course, had these points been thrown out earlier, it may have been the case later, due to changes in vehicle orientation, that too many or too few points had been thrown out). Every iteration, STRIPE estimates the length of the path ahead of the vehicle's origin. As soon as the length of the path ahead is estimated to be less than three meters, all of the points ahead of the vehicle are thrown out.

Holding on to the points until the last minute has the additional advantage of ensuring that the vehicle orientation before it reaches the end of the path is as close as possible to the orientation specified by all of the points.

## 3.3 Reprojection Details

STRIPE needs to keep track of certain coordinate frame transformations in order to do the incremental reprojection of a path created at an old position onto a new ground plane. In particular, in order to project an image taken in a previous location to the groundplane of the vehicle's current location, we need to know the transformation between the old camera's coordinate system

and the current vehicle coordinate system. In this discussion the following notation is used: the transformation between $\alpha$ and $\beta$ is denoted by $T_{\alpha}^{\beta}$. The equation $q = \left(T_{\alpha}^{\beta}\right)p$ transforms the point p (currently in $\alpha$'s coordinate frame) to a point q in $\beta$'s coordinate frame.

The row and column location of the 2D image waypoint is known. This is turned into a 3D point in camera coordinates using the row and column factors of the camera (see Appendix A). The camera focus is defined to be the origin of the camera coordinate system, and the distance to the image plane is 1 (the row and column factors are scaled by the focal length).

The transformation between the vehicle coordinate frame and its corresponding camera coordinate frame at the time the image was digitized will be referred to as $T_{v\_old}^{c}$.

Assume that we know the transformation between some world coordinate frame, w, and the vehicle position when the image was taken ($T_{w}^{v\_old}$), and the transformation between the same world coordinate frame and the current vehicle location ($T_{w}^{v\_current}$). Then,

$$T_{v\_current}^{c} = \left(T_{v\_old}^{c}\right)\left(T_{w}^{v\_old}\right)\left(T_{w}^{v\_current}\right)^{-1} \tag{3.1}$$

$\left(T_{v\_current}^{c}\right)^{-1}$ can be used to compute the location of the camera focus and the image point in current vehicle coordinates. The vehicle groundplane is at $z = 0$ in current vehicle coordinates, and the intersection of the line between the camera focus and the image point, with the current vehicle groundplane, is location of the projection of that image point.

Note that relative position information, such as data from an INS, is sufficient for the STRIPE system. No global positioning information is necessary; the actual location of the origin of the world coordinate frame is irrelevant. Absolute position is irrelevant because STRIPE only needs to know the relative transformation between the original position (when the image was taken) and the current position.

## 3.4 The Initial User Interface

The initial design of STRIPE concentrated on the vehicle side of the problem, and little thought was given to the design of the user interface. It was assumed that because human ability to interpret images is far superior to machine ability, the system would be easy to use, even with a

very minimalistic operator user interface. Figure 3.9 shows the initial user interface, which consisted of three components: the image window, the control window, and the message window.

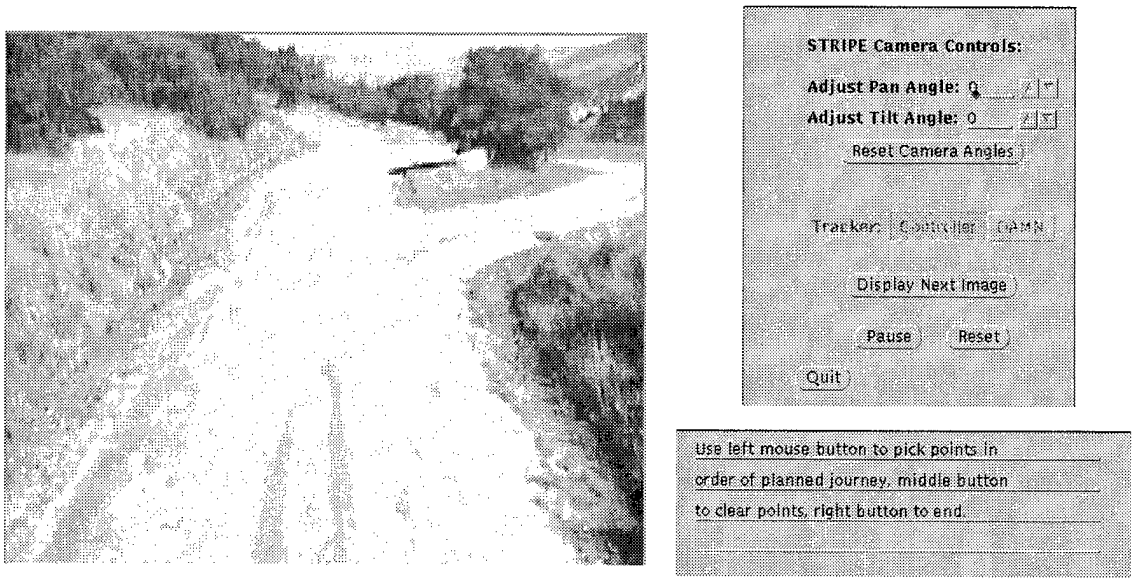



**Figure 3.9 The original operator interface**

The image window, measuring about 18x12 cm on the workstation monitor, displayed the current image transmitted from the vehicle, and was where the operator chose waypoints. The control window allowed the operator to adjust the camera pan and tilt angles, stop and restart the vehicle, and request a new image. Finally, the message window provided text messages for the operator to provide hints as to the operators task.

After initialization, the operator's workstation was in *pick points* mode. The operator waited for an image to appear on the image window, and then picked points by moving the cursor to the desired location in the image and pressing the left mouse button. A small marker appeared under the cursor. If a mistake was made while picking points, pressing the middle mouse button erased all of the image waypoints (and blacked out their corresponding blobs) and the operator could pick new points. When the operator finished picking waypoints, the right mouse button was pressed, the points are transmitted to the vehicle, and the procedure begins again.

If, for some reason, the operator could not pick points in the current image, he or she had to exit pick points mode by pressing the right mouse button. Once out of pick points mode, the oper-

ator could use the control panel to adjust the pan and tilt angles of the camera, stop (pause) and start (reset) remote vehicle motion, and request a new image (display next image). Pressing the *display next image* button also returned the operator to pick points mode.

## 3.5 Implementation Details

Testing is performed using the Carnegie Mellon Navlab 2 [40], a U.S. Army HMMWV Ambulance that has been reconfigured to be completely computer controllable (see Figure 3.10).

**Figure 3.10 The Carnegie Mellon Navlab 2**

The STRIPE vehicle modules run on a Sun Sparcstation 10 mounted on Navlab 2, and the vehicle controller runs on a 68000-based computer. The main STRIPE reprojection loop can run at up to 80 Hz on a Sun Sparcstation 10, but has to be slowed down to 5 Hz on Navlab 2 to avoid overloading the vehicle controller.

Positioning information is provided by a Modular Integrated Avionics Group Advanced Development Module (MIAG ADM) navigation and guidance sensor manufactured by the Lear Astronics Corporation. Local positioning information is computed via 3-D dead reckoning using attitude, heading, and odometer, and has a resolution of 0.00102 meter, and an accuracy of one

percent of distance travelled. Roll, pitch and yaw have a resolution of 0.00275 degrees. Roll and pitch are accurate to 0.4 degrees, and yaw is accurate to 2 degrees.

A Sony XC75 black and white video camera with an 8mm lens is mounted on a Remote Ocean Systems pan/tilt unit centered on the outside of the vehicle just above the roof, at a height of about 2.1 meters off the ground. Images were captured onto a datacell color video digitizer. The pan/tilt unit is limited to about 72 degrees of pan and 30 degrees of tilt to prevent collisions with other equipment mounted on the vehicle.

The operator controls run on a Sparc LX class portable workstation, with the display measuring approximately 21.5 cm x 16.5 cm. The display is less than ideal. It is fairly dim, susceptible to significant glare, and displays only 8 graylevels in the monochrome images.

Communication between the operator workstation and the STRIPE vehicle is achieved using a pair of Aironet Arlan 640 Wireless Ethernet Bridges with supplementary omnidirectional antennas. These bridges provide line-of-site coverage at up to 380 meters between units, with a bandwidth of up to 230 Kbits/second. Some initial experimentation was done with a pair of cellular modems, but the bandwidth and ease of use of the Arlan bridges proved to be far superior.

Inter-module communication between the workstations as well as within the workstations was achieved using the IPT toolkit [14] with additional stubs from the ALVINN [35] libraries. Optional image compression was provided using JPEG [41] compression software which was a minor modification of release 4 of the Independent JPEG Group's free JPEG software.[2] The ALVINN tracker library provided 3D waypoint tracking.

On average, 12.9 seconds elapsed between the time the operator clicked the right mouse button to send the waypoints to the vehicle and the time that the next image appeared on the operator's monitor. In addition to the time spent transmitting the data over the wireless ethernet, this value includes overhead due to digitizing, waiting to get current position information from the vehicle controller (at least 200 ms), JPEG compression, inter-process communication, JPEG

2. Available from ftp.uu.net in graphics/jpeg

decompression, display of graphical pan and tilt values, and display of the image on the operator monitor.

STRIPE was also implemented at Lockheed Martin in Denver, Colorado using similar equipment.

# Chapter 4 Error Analysis

## 4.1 Overview

In STRIPE, each pixel in the operator's image corresponds to a patch of ground in the real world that is approximately trapezoidal. A change in the shape of the terrain can cause an error in the predicted real-world location of a given pixel. In a theoretical STRIPE system with perfect sensors and perfect calibration, this error becomes almost non-existent as the vehicle approaches the real-world patch of ground. However, few sensors are perfect, and the calibration method used for this system (see Appendix A) is unlikely to produce a perfect result. Thus it is important to consider how errors in the system affect the mapping from pixel to ground patch. Several of the potential errors produce the same type of effect on the predicted location of a waypoint, and this chapter is organized into sections corresponding to those classes of errors.

## 4.2 Sources of Error

The six sources of error that are considered in this chapter are: camera calibration, vehicle-to-camera transformation, physical terrain, inertial sensor data, limited resolution, and human errors.

The effects of problems such as chromatic aberration, lens distortion, and sensor misalignment are thesis topics in their own right, and will be left to the calibration experts.

### 4.2.1 The Six Sources of Error

*Camera Calibration*

STRIPE camera calibration (described in Appendix A) uses a pinhole-camera model for its reprojections. The purpose of the calibration is to determine the row and column factors of the camera. How does an error in the row or column factor affect the corresponding real world location of a projected pixel?

*Vehicle to Camera Transformation*

To perform STRIPE reprojection, the system needs to know the location of the camera in the vehicle's coordinate frame. How do errors in the vehicle to camera transformation affect the predicted location of image points in the real world?

*Terrain*

STRIPE continuously approximates the location of a given waypoint on the ground by reprojecting that waypoint onto the current groundplane. Some time before we arrive at the point on the ground, we must use its estimated location to compute a steering angle. What sort of errors does this produce?

*Inertial Data*

In order to accurately reproject the image points onto the groundplane, STRIPE depends heavily on its knowledge of the location of the vehicle. However, an error in the inertial data is relevant only if it accumulates rapidly. Each time a new STRIPE image is digitized, the error is effectively reset to zero. This is because the STRIPE reprojections are all relative to the location of the vehicle when the image for those waypoints was digitized. Thus, errors accumulate only over a very short distance (on the order of 10's of meters) and are therefore almost negligible. Those errors that do exist have the same effects as the errors in the vehicle to camera transformation.

It is important to consider both the resolution and the accuracy of the measurements produced by the sensor. The resolution of our sensor, i.e., the smallest discernible reported unit, is 0.00102 meters for x, y, and z local position, and 0.00275 degrees for roll, pitch, and yaw. The accuracy, i.e., the degree to which the information matches true or accepted values is 1% of distance travelled for x, y, and z local position, 0.4 degrees for roll and pitch, and 2.0 degrees for yaw. How does an error in the reported value of the inertial data affect the reprojection of the STRIPE points?

*Error Due To Limited Resolution*

A pixel in the image corresponds to a patch on the ground that is approximately trapezoidal. As you move up the y-axis of the image plane, a pixel represents a larger and larger patch of ground. How large do these trapezoids become?

*Human Error*

An integral part of the STRIPE system is the human operator, who chooses the 2D image points. The human introduces two types of errors. First, what is the magnitude of the error in the location of a waypoint when a human operator picks a point a pixel or two away from the intended one in the image? An analysis of the effects of this sort of error are presented in this chapter.

The other type of human error is more difficult to quantify. How often is a human confused by the image that is presented, and picks a point that has no relation to where he or she wishes the vehicle to go. The user studies described in Chapter 6 provide insight into this question.

## 4.3 Coordinate Frames

Figure 4.1 and Figure 4.2 show the coordinate frames used throughout this chapter. Figure 4.1 shows the vehicle coordinate frame, the origin of which lies on the ground directly below the center of the real axle. Figure 4.2 shows the camera coordinate frame, which has its origin at the camera focus, as viewed from the side of the camera, as well as the projection of the coordinate frame onto the image plane as viewed from in front of the vehicle.

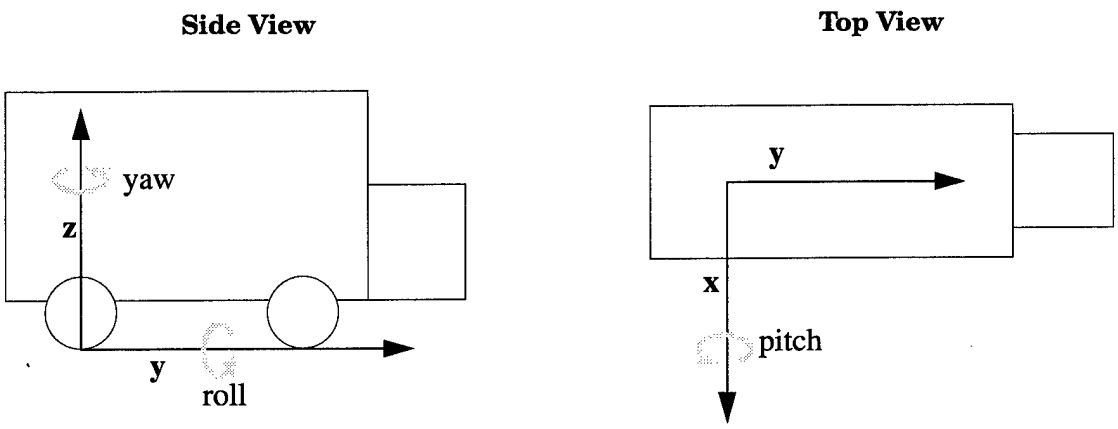



**Figure 4.1 The vehicle coordinate frame, as viewed from the side and from above.**

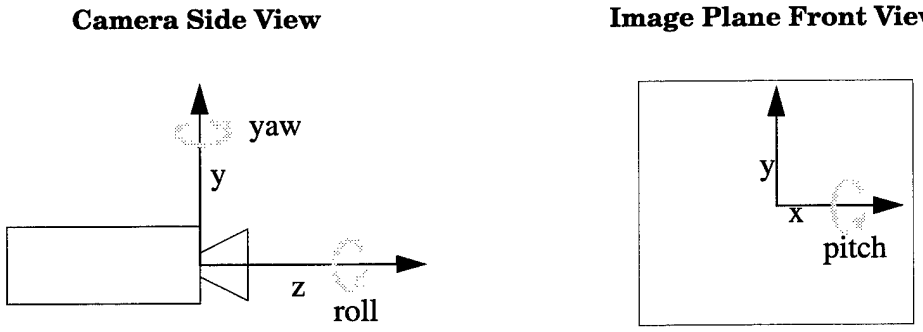



**Figure 4.2 The camera's coordinate frame, as viewed from the side of the camera, and in front of the camera on the image plane.**

## 4.4 Assumptions

Many of the following sections contain graphs and values of the errors for a scenario roughly based on the typical STRIPE setup. Unless otherwise noted, these assume that:

The camera has a focal length of 10 millimeters.

The camera has zero roll or yaw, and has a pitch of -15 degrees.

The camera is located at z=3 meters in vehicle coordinates.

The human intends to pick the optimal pixel in the image for the given task.

The camera has a vertical field of view of 35 degrees, and 500 rows.

The camera has a horizontal field of view of 42 degrees, and 600 columns.

Note that the last two assumptions mean that the width of a pixel is different from its height. In the sections that follow, the term "vertical pixel", or "vpixel", will be used to describe the height of a pixel, and "horizontal pixel", or "hpixel", will be used to describe a pixel's width.

# 4.5 Errors Along The Vehicle's x, y, and z Axes

## 4.5.1 Errors Along The Vehicle's x and y Axes

Errors along the x and y axes of the vehicle have a constant effect on the projected position of the waypoint on the groundplane. Figure 4.3 shows the case where the calibration shows the camera to be a distance e further forward along the x axis than it really is. Simple geometry shows that this puts the actual location of the projected point at a distance **e** closer to the vehicle along the x axis than expected.

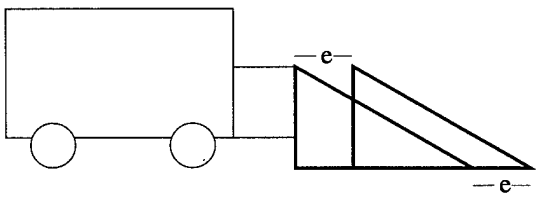



**Figure 4.3 Error in the x vehicle to camera translation.**

Similarly, Figure 4.4 shows the case where the calibration puts the camera a distance e to the left of the actual camera origin. In this situation, the actual waypoint lies at a distance e to the right (i.e., along the y axis in vehicle coordinates) on the ground from the predicted location.

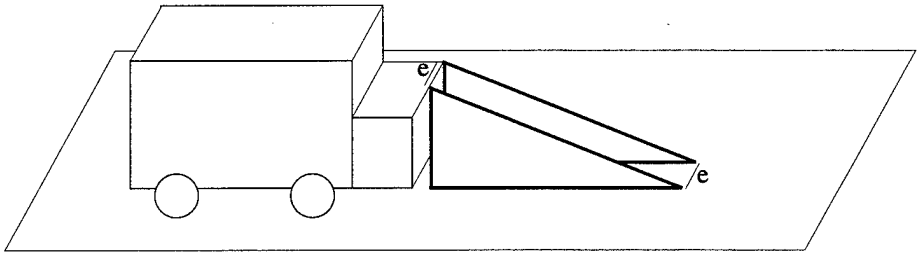



**Figure 4.4 Error in the y vehicle to camera translation.**

Clearly an error in the calibration of either one of these components, even one as large as 10 or 20 centimeters, would have little effect on the accuracy of the system as a whole.

Similarly, the error due to inertial sensor accuracy is reset to zero every time a new image is digitized. Over a 30 meter course one would expect the x error to accumulate to no more than about 30 centimeters, again not too significant a deviation.

### 4.5.2 Errors along the vehicle's z axis

What is the effect of an error along the vehicle's z axis? Consider Figure 4.5.

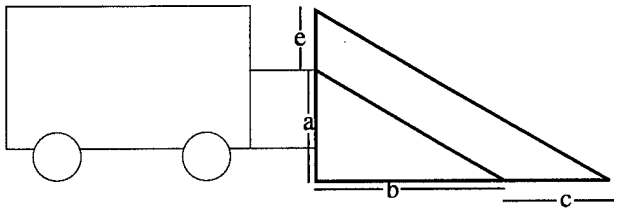



**Figure 4.5 Error in the z vehicle to camera translation.**

$$\frac{a}{a+e} = \frac{b}{b+c} \tag{4.1}$$

$$b+c = \frac{b\,(a+e)}{a} \tag{4.2}$$

$$\frac{c}{e} = \frac{b}{a} \tag{4.3}$$

What kind of error due to calibration can we expect to see in practice? Figure 4.6 shows the magnitude of c, the error along the vehicle's y axis, for different values of b and e. Figure 4.7 shows the magnitude of c when e is fixed at 5 centimeters.

## 4.6 Errors along the image plane

What happens if the computed location of a 2D point on the image plane, along one or both axes, is incorrect? As will be shown in sections 4.6.1 and 4.6.2, errors parallel to the image plane's x axis produce errors on the groundplane parallel to the vehicle's x axis, and errors parallel to the image plane's y axis produce groundplane errors parallel to the vehicle's y axis.

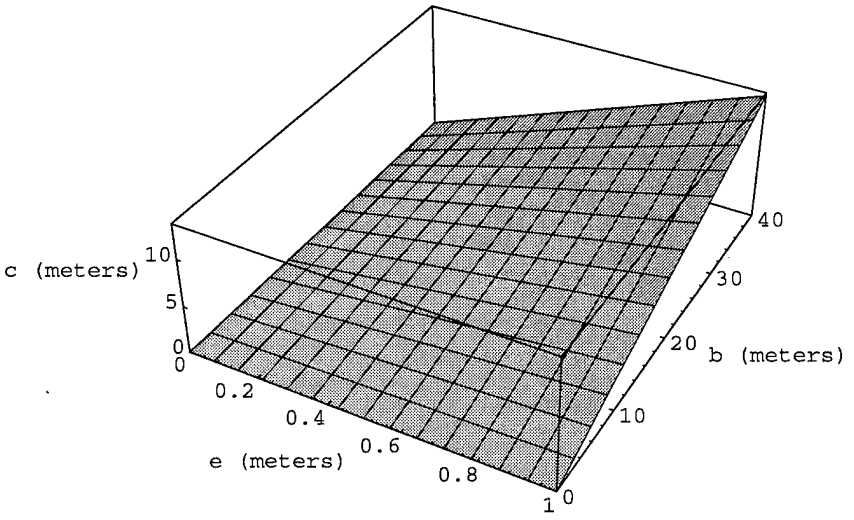



**Figure 4.6 Error due to a vehicle to camera calibration error in z.**

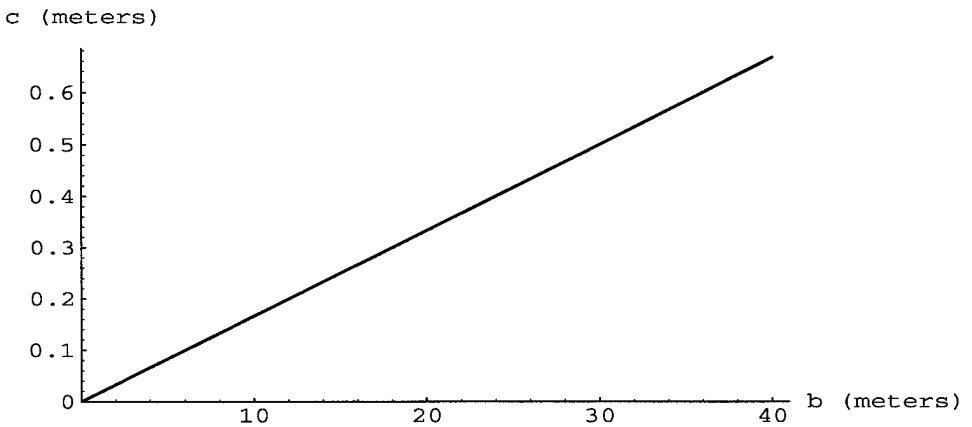



**Figure 4.7 Error due to a vehicle to camera calibration error in z of 5 centimeters.**

## 4.6.1 Errors parallel to the y axis of the image plane

### 4.6.1.1 Derivation

Consider Figure 4.8. The camera focus is at point A, and the image plane lies along line $\overline{BD}$ (thus f, the camera focal length, is the length of line segment $\overline{AB}$). The camera is tilted at an angle $\frac{\pi}{2} - \beta$ from the horizontal. The chosen point actually lies at a distance b above the center of the image plane, but the predicted image point is an additional error e further up the image plane.

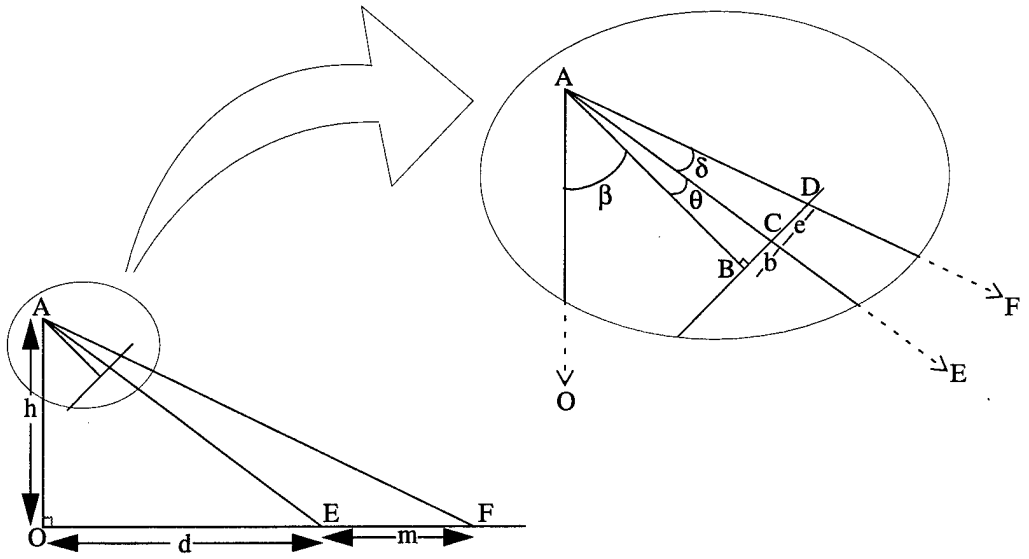



**Figure 4.8 Error along the y axis of the image plane**

Problem: compute m, the additional distance along the ground plane the error will introduce, in terms of:

- h: height of the camera above the ground
- b: correct vertical distance along image plane
- e: additional vertical distance along image plane due to row factor error
- f: focal length = length of line segment AB.
- β: 90 - tilt of the camera ("90 minus tilt of the camera")

The result is a rather complex and relatively unrevealing equation. Rather than presenting the full equation, the algorithm for computing m is presented:

$$\theta = \operatorname{atan}\frac{b}{f} \tag{4.4}$$

$$\tan(\theta + \delta) = \frac{b+e}{f} \tag{4.5}$$

$$\delta = \operatorname{atan}\left(\frac{b+e}{f}\right) - \theta \tag{4.6}$$

$$d = h \cdot \tan(\beta + \theta) \tag{4.7}$$

$$d + m = h \cdot \tan(\beta + \theta + \delta) \tag{4.8}$$

$$m = h(\tan(\beta+\theta+\delta) - \tan(\beta+\theta)) = h\left(\frac{\sin((\beta+\theta+\delta)-(\beta+\theta))}{\cos(\beta+\theta+\delta)\cos(\beta+\theta)}\right) \quad (4.9)$$

$$m = h\left(\frac{\sin(\delta)}{\cos(\beta+\theta+\delta)\cos(\beta+\theta)}\right) \quad (4.10)$$

#### 4.6.1.2 Effect

Despite the ugliness of the equations, some more straightforward information can be gleaned from simply considering the situation.

The most important variable is the sum of β and θ. As β+θ approaches 90 degrees, m, no matter how small e may be, approaches infinity. As β+θ becomes smaller, the same value of e produces decreasing values for m. The STRIPE scenario reduces the likelihood that β+θ would be near 90 degrees, since there does not tend to be sufficient details in features near the horizon line for a point to be chosen there.

Figure 4.9 shows the error along the ground (m) for some reasonable distances (d), given an error of e vertical pixels for our standard setup described in Section 4.4.

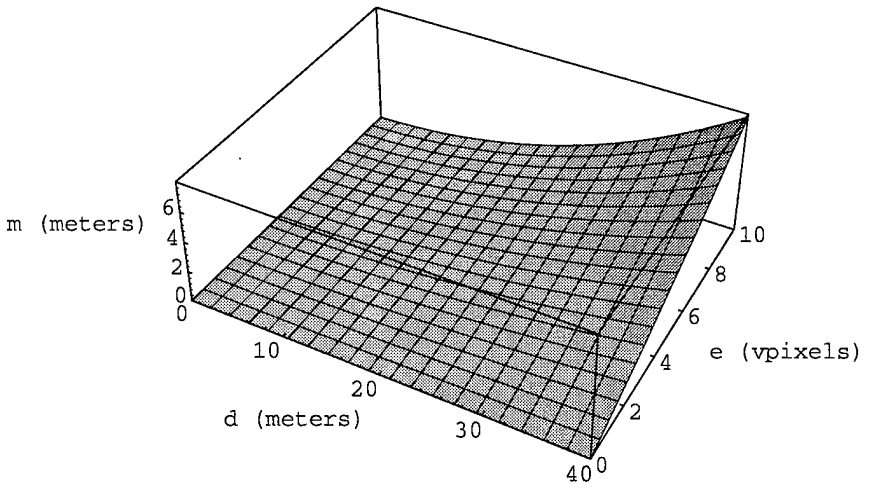



**Figure 4.9 Error along the y axis of the image plane.**

#### 4.6.1.3 Conditions

An error along the y axis of the image plane can occur because of human error in point picking, and because of an error in the computed row factor.

*Human Error*

A human's ability to accurately pick a point is independent of the location of that point in the image. Figure 4.10 shows the effect of an error of one pixel along the y axis.

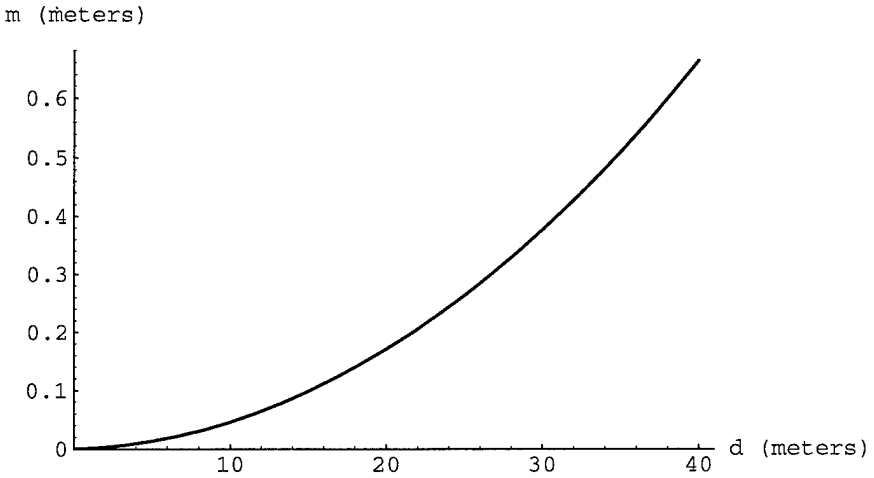



**Figure 4.10 An error of one vpixel along the y axis of the image**

*Error Due To Limited Resolution*

Figure 4.10 can also be interpreted as the longitudinal size of the projection of a pixel on the groundplane as one moves up the image plane.

*Error in Row Factor*

The row factor is the distance between two adjacent row centers, scaled by the focal length. An error in row factor accumulates as the absolute value of the y position on the image plane increases. Thus, as the length of b in Figure 4.8 increases, so will e. This effect is demonstrated in Figure 4.11. The graph assumes that there is an error of $\frac{4}{500}$vpixel in row factor, i.e., at the image center the error due to row factor is (e) 0, 10 pixels up from the image center e is $\frac{4}{50}$vpixel, and at the top of the image, e is $2 \cdot$ vpixel. Note that Figure 4.11 does not include values for d less than about 12, because when d is smaller the ray AE passes through the lower half of the image.

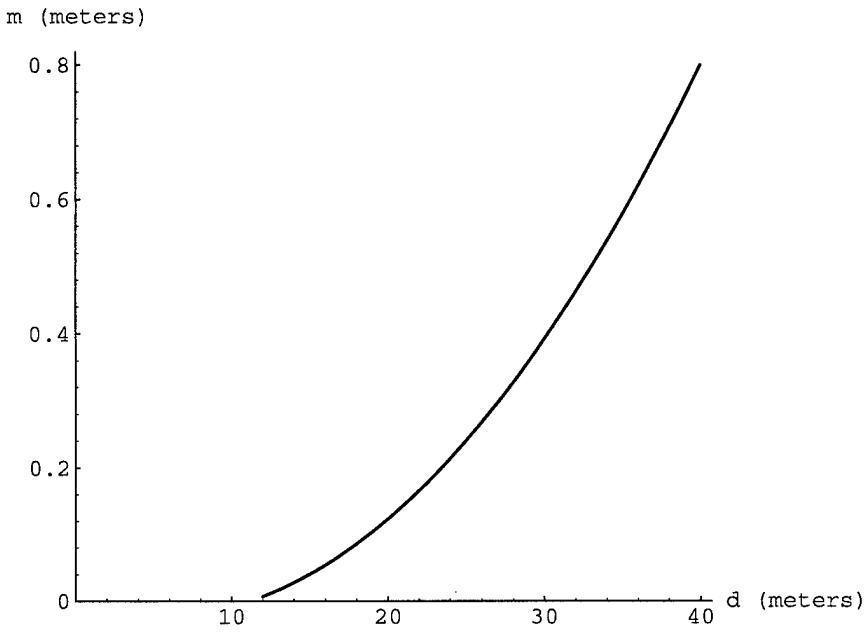



**Figure 4.11 Error in row factor.**

*Error in the vehicle to camera pitch*

An error in the vehicle to camera pitch of $\lambda$ (see Figure 4.12) is equivalent to assuming that $\delta$ in Figure 4.8 is known and has a value of $\lambda$ (thus eliminating the need to know the value of e in that computation). Figure 4.13 and Figure 4.14 show how pitch and distance affect the location of a waypoint on the ground.

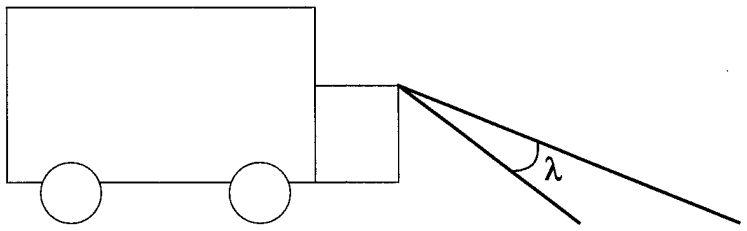



**Figure 4.12 Error in the vehicle to camera pitch**

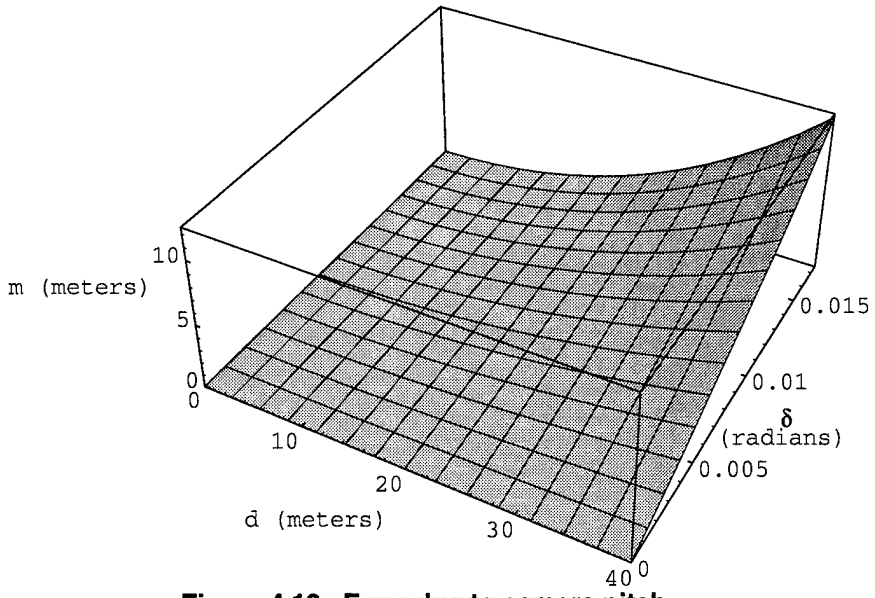



**Figure 4.13 Error due to camera pitch.**

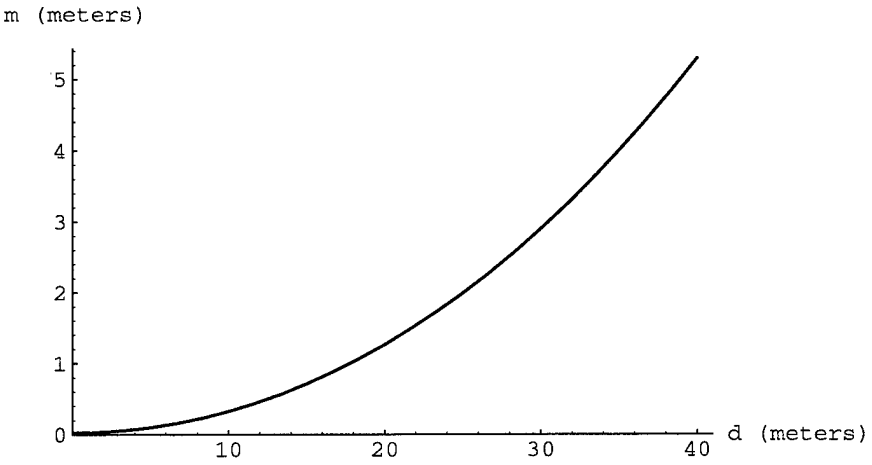



**Figure 4.14 Error due to a pitch of 0.5 degrees.**

*Error in the inertial sensor pitch*

The accuracy of the inertial sensor is a 0.57 degree angle in pitch, very close to that depicted in Figure 4.14.

## 4.6.2 Errors parallel to the x axis of the image plane

### 4.6.2.1 Derivation

Consider Figure 4.15, which is an extension to Figure 4.8. Those angles and points with the same label represent the same object. The camera's focus is still at point A, and a three-dimensional representation of the image plane is presented. H is the actual point chosen, and the ray $\overrightarrow{AH}$ intersects the groundplane at point J. However, due to an error in the column factor, we are actually projecting out along the ray $\overrightarrow{AG}$.

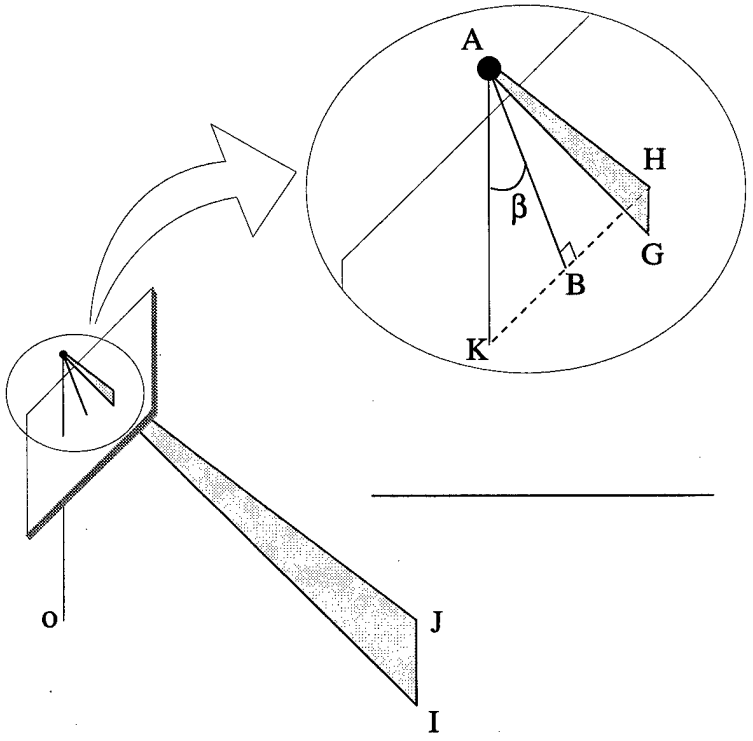



**Figure 4.15 Error along the x axis of the image plane.**

Problem: compute the length of $\overline{IJ}$, the error along the ground due to the incorrect column factor, in terms of:

- $\overline{BH}$: vertical distance along the image plane
- $\overline{GH}$: horizontal distance along the image plane due to column factor error
- $\beta$: tilt of the camera

- f: focal length = length of $\overline{AB}$
- h: height of the camera above the ground = length of $\overline{OA}$

$$\overline{AH}^2 = f^2 + \overline{BH}^2 \tag{4.11}$$

$$\angle BAH = \text{atan}\frac{\overline{BH}}{f} \tag{4.12}$$

$$\overline{OJ} = h \cdot \tan(\beta + \angle BAH) \tag{4.13}$$

$$\overline{AJ}^2 = h^2 + \overline{OJ}^2 \tag{4.14}$$

$$\frac{\overline{AH}}{\overline{AJ}} = \frac{\overline{GH}}{\overline{IJ}} \tag{4.15}$$

$$\overline{IJ} = \frac{\overline{GH} \cdot \overline{AJ}}{\overline{AH}} \tag{4.16}$$

### 4.6.2.2 Effect

In practice, the effect of an error along the x axis of the image plane has a significantly smaller effect than that of an error along the y axis (see Figure 4.16).

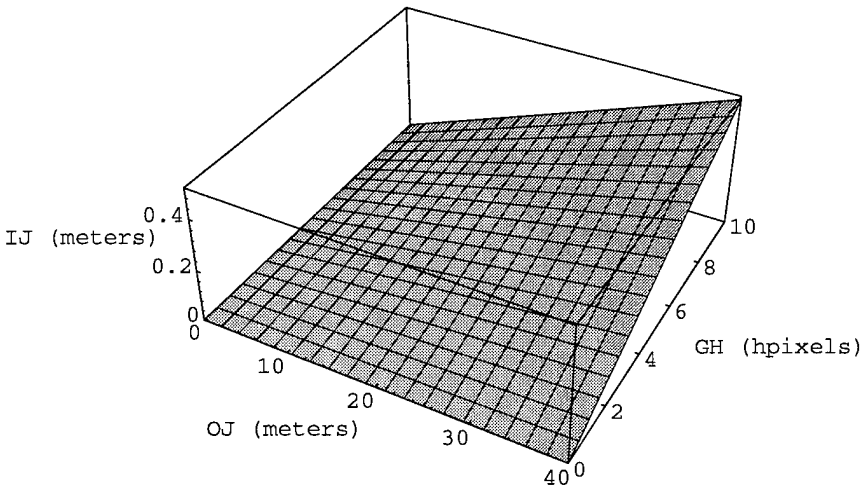



**Figure 4.16 Error along the x axis of the image plane.**

*Human Error*

Figure 4.17 shows the effect of an error of one hpixel along the x axis of the image plane

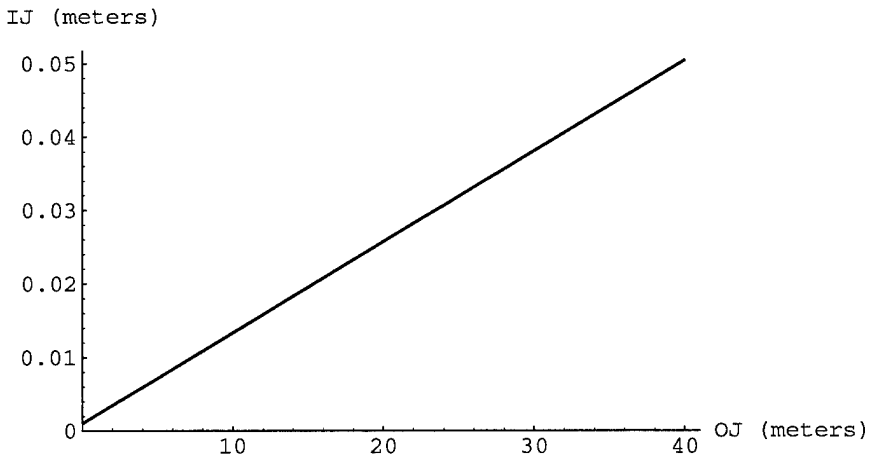



**Figure 4.17 An error of one hpixel along the x axis of the image plane**

*Error Due To Limited Resolution*

Figure 4.17 can also be interpreted as the horizontal size of the projection of a pixel on the groundplane as one moves up the image plane.

*Error in Column Factor*

The column factor accumulates as the absolute value of the x position on the image plane increases. Figure 4.18 shows this effect, with an error of $\frac{4}{600}$hpixel in column factor (meaning an error at the right side of the image is $2 \cdot$ hpixel

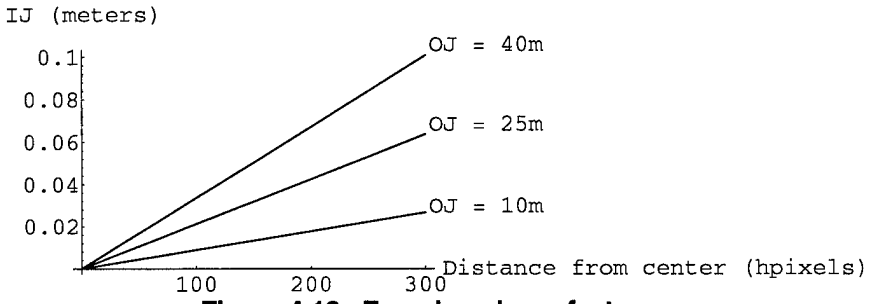



**Figure 4.18 Error in column factor.**

### 4.6.3 Errors about both axes of the image plane

An error in the vehicle to camera roll due to calibration or the inertial sensor has an effect on any given point similar to that of an error in both the row and column factor. Simply use $\Delta x$ for the magnitude of $\overline{GH}$ in Figure 4.15 and $\Delta y$ for the magnitude of e in Figure 4.8

$$\Delta y = r\sin(\xi + \eta) - r\sin\eta \tag{4.17}$$

$$\Delta x = r\cos(\xi + \eta) - r\cos\eta \tag{4.18}$$

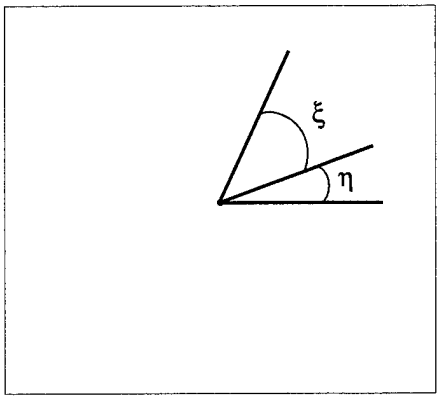



**Figure 4.19 Error in Vehicle to Camera Roll**

Figure 4.20 shows the effect of an error in roll for the $\Delta y$ component of roll, for various values of $\zeta$ and $\eta$. Figure 4.21 shows how the projection of the $\Delta y$ component behaves when $\eta = 0$. Figure 4.20 is actually the result of taking the difference between Figure 4.21 and itself offset by $\eta$. Figure 4.22 and Figure 4.23 show the corresponding effect for the $\Delta x$ component.

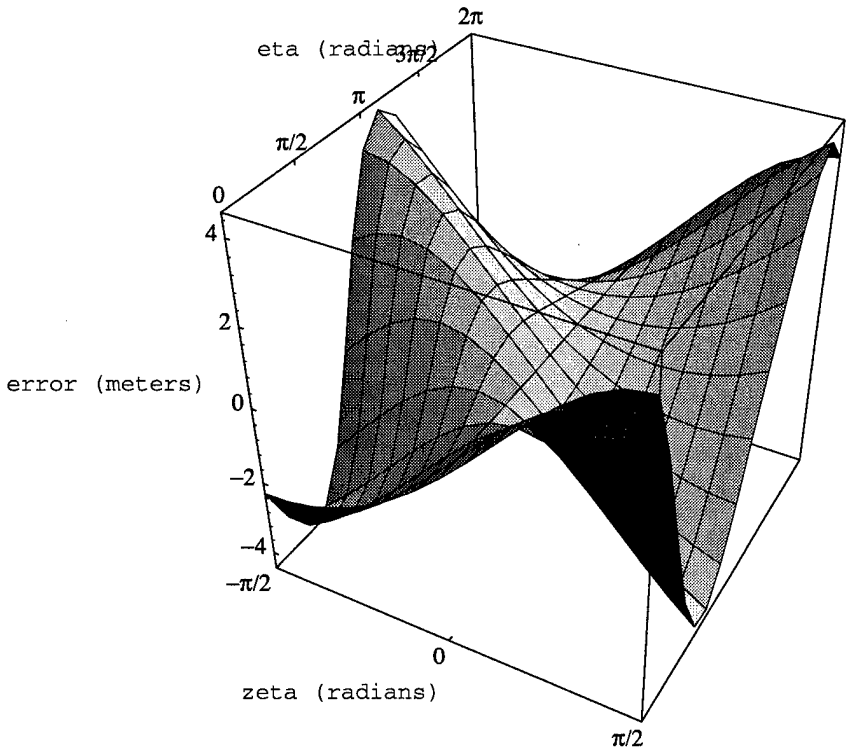



**Figure 4.20 Error in roll: y axis component at 50 vpixel radius.**

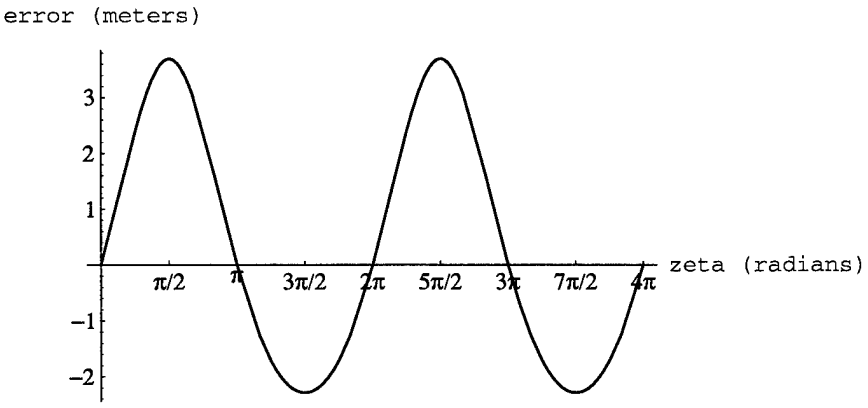



**Figure 4.21 Error in roll: y axis component, r= 50 vpixel, η=0**

# 4.7 Other Errors

## 4.7.1 Error in the Vehicle to Camera Yaw

An error of $\alpha$ in the vehicle to camera yaw (Figure 4.24) due to either calibration or inertial error results in an error in the x and y position of the projected point on the ground

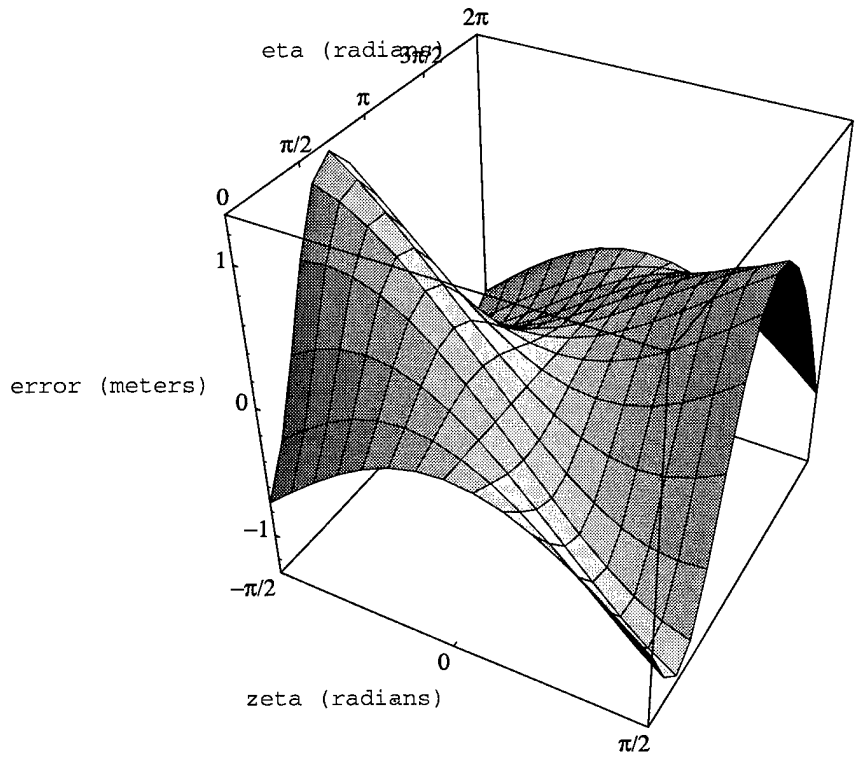



**Figure 4.22 Error in roll, x axis component at 50 vpixel radius.**

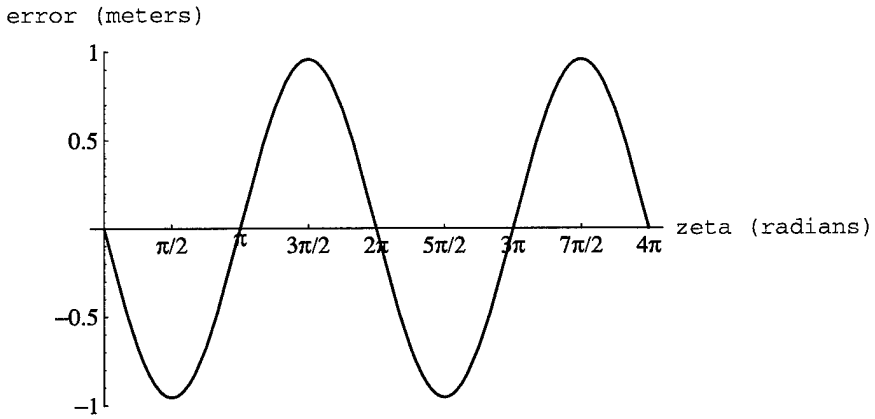



**Figure 4.23 Error in roll, x-axis component, r = 50 vpixel, η=0**

$$\text{error in x} = d\sin(\alpha+\tau) - d\sin\tau \tag{4.19}$$

$$\text{error in y} = d\cos\tau - d\cos(\alpha+\tau) \tag{4.20}$$

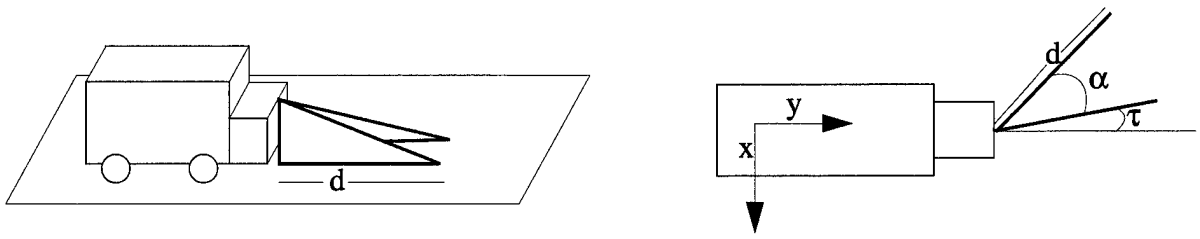



**Figure 4.24 Error in the vehicle to camera yaw.**

Figure 4.25 and Figure 4.26 show the X component of the error at d= 25m, and Figure 4.27 and Figure 4.28 show the Y component of the error.

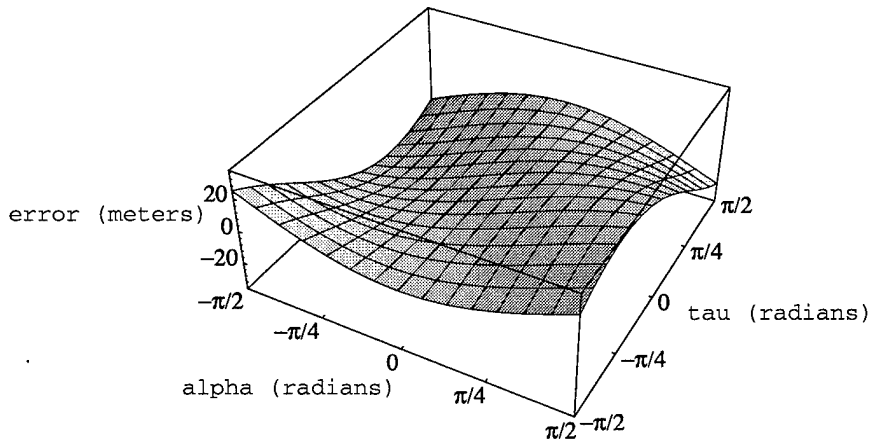



**Figure 4.25 X component of error in vehicle to camera yaw at a d = 25 m.**

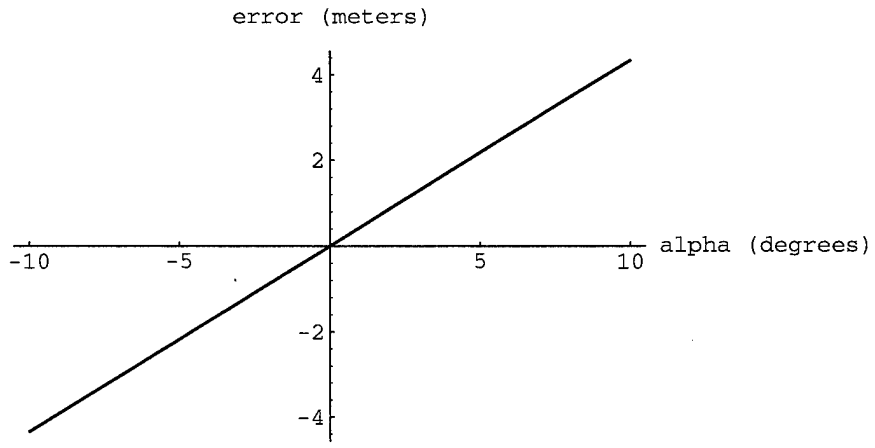



**Figure 4.26 X component of Error in vehicle to camera yaw at d=25m, τ=0**

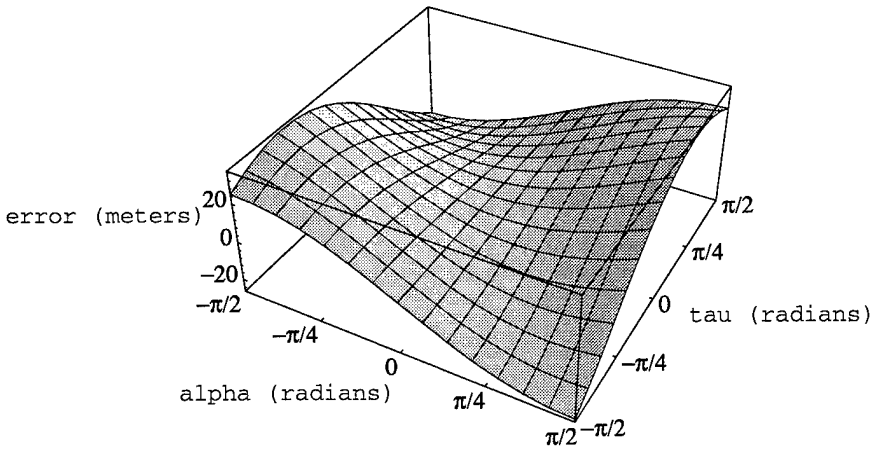



**Figure 4.27 Y component of error in vehicle to camera yaw at d = 25m.**

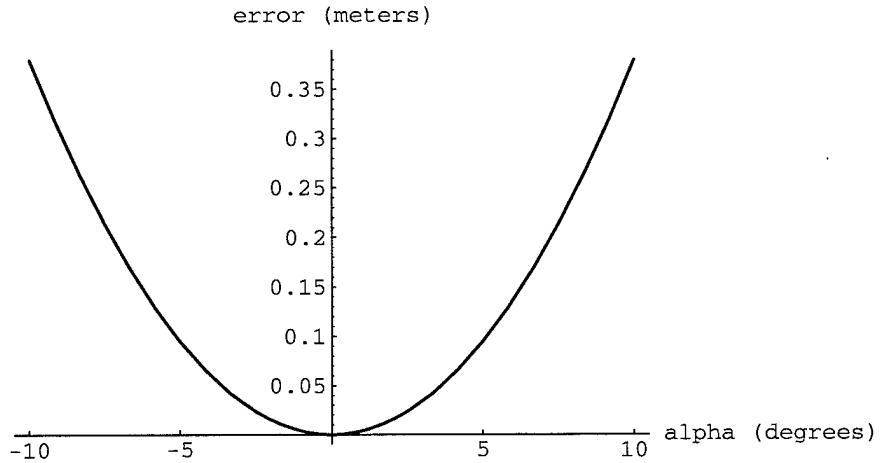



**Figure 4.28 Y component of Error in vehicle to camera yaw at d=25m, τ=0**

## 4.7.2 Errors Due to Varying Terrain.

STRIPE continuously approximates the location of a given waypoint on the ground by reprojecting that waypoint onto the current groundplane. Sometime before we arrive at the point on the ground, we must use its estimated location to compute a steering angle. If the groundplane under the vehicle does not match the plane of the point, there will be an error in this estimate (see Figure 4.29). The camera is located at point A, and we are considering the projection of a pixel that

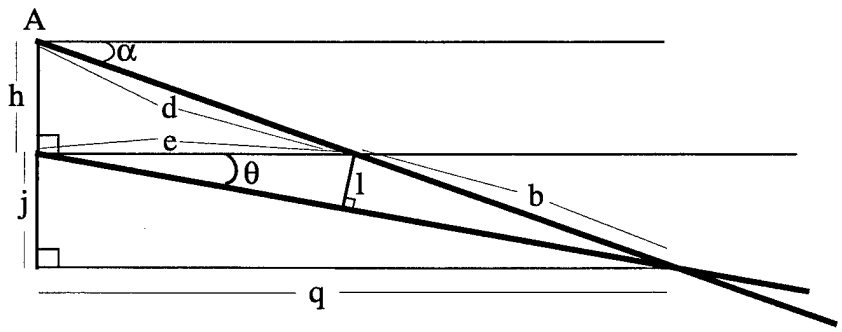



**Figure 4.29 Errors due to varying terrain**

makes an angle $\alpha$ with the horizontal. The vehicle sits on the horizontal plane e and immediately in front of the vehicle, the groundplane changes, and makes an angle of $\theta$ with the horizontal. Problem: What is $\frac{q}{e}$, the ratio between the erroneous distance computed and the actual distance, given:

$\alpha$: the angle the projection of the pixel makes with the horizontal

$\theta$: the angle that the new groundplane makes with the current one

h: the height of the camera off the ground

$$e = h \cdot \cot\alpha \tag{4.21}$$

$$l = e\sin\theta = b\sin(\alpha - \theta) \tag{4.22}$$

$$b = \frac{e\sin\theta}{\sin(\alpha - \theta)} \tag{4.23}$$

$$q = (b + d) \cdot \cos\alpha \tag{4.24}$$

An alternative way to consider this is as the intersection of two line segments, that intersect when $y_1 = y_2$:

$$y_1 = \frac{-h}{e}x + h = h - x\tan\alpha \qquad y_2 = \frac{-j}{q}x = -x\tan\theta \tag{4.25}$$

$$h - x\tan\alpha = -x\tan\theta \tag{4.26}$$

$$x = q = \frac{h}{\tan\alpha - \tan\theta} \tag{4.27}$$

$$e = \frac{h}{\tan\alpha} \tag{4.28}$$

$$\text{error} = q - e = \frac{h}{\tan\alpha - \tan\theta} - \frac{h}{\tan\alpha} \tag{4.29}$$

Figure 4.30 shows how the error due to change in terrain is related to the orientation of the terrain relative to the vehicle's groundplane and the distance to the projected point on that groundplane. Figure 4.31 shows that same error for a fixed distance of 5 meters to the projected point on the ground.

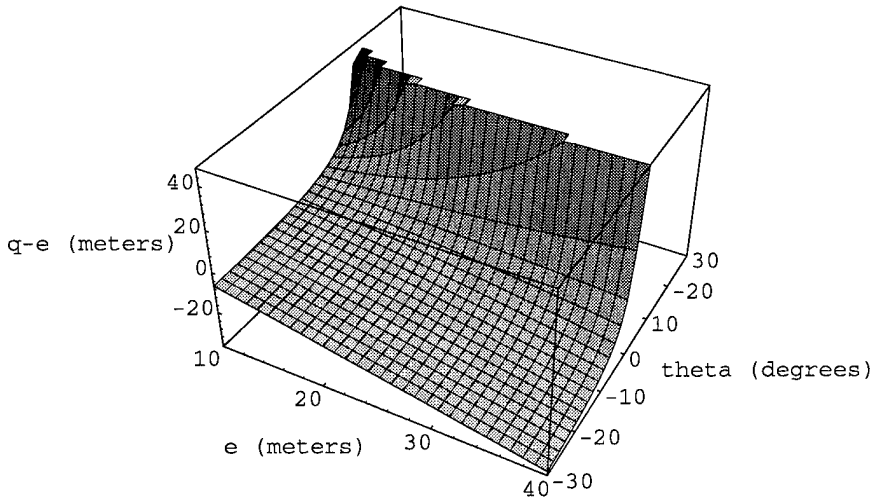



**Figure 4.30 Error due to change in terrain.**

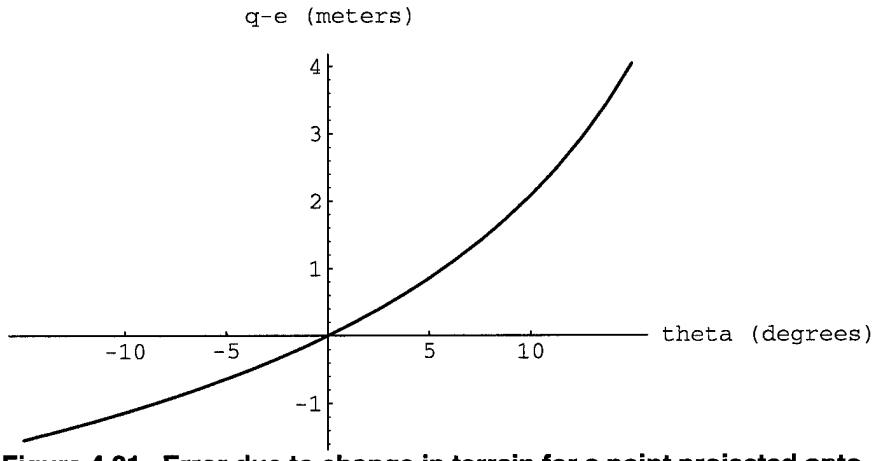



**Figure 4.31 Error due to change in terrain for a point projected onto the vehicle's groundplane at 5m.**

# 4.8 Combinations of Errors

Most of the errors considered here have an effect of moving the projection ray along the x or y axis of the image plane, as shown in Figures 4.8 and Figures 4.15. In the worst case, errors on the y axis (similarly the x axis) of the image plane would sum together to provide a combined error in the projection ray.

## 4.8.1 Summation of errors along the y axis of the image plane

The errors along the y axis of the image plane are detailed in Section 4.6.1 and 4.6.3, and consist of errors due to human error, limited resolution, row factor, vehicle to camera pitch, and vehicle to camera roll. Thus the worst error along the y axis of the image plane is:.

$$\begin{aligned}\text{Worst error along x axis} = {} & (\text{Human error in pixels} \times \text{vpixel}) \\ & + (1 \times \text{vpixel}) \\ & + (\text{Row factor error} \times \text{Vertical distance from image center}) \\ & + (\text{Pitch error along image plane}) \\ & + \text{Roll error along the y axis}\end{aligned} \tag{4.30}$$

All of the quantities on the right hand side of Equation (4.30) are measured along the y axis of the image plane in the previous derivations, except for the pitch error, which is usually thought of as an error in angle, rather than a physical distance along the image plane. From Figure 4.8 and Equations (4.4) and (4.5), we can derive the pitch error along the image plane:

$$\text{Pitch error along image plane} = f \times \tan(\theta + \delta) - b \tag{4.31}$$

Given specifications for particular sensors, a worst case error along the y axis of the image plane could be computed using Equations (4.30) and (4.31) together with the algorithm presented in Equations (4.4) through (4.10).

### 4.8.2 Summation of errors along the x axis of the image plane

The errors along the x axis of the image plane are detailed in Section 4.6.2 and 4.6.3, and consist of errors due to human error, limited resolution, column factor, and vehicle to camera roll. Thus the worst error along the x axis of the image plane is:

$$\begin{aligned} \text{Worst error along x axis} = {} & (\text{Human error in pixels} \times \text{hpixel}) \\ & + (1 \times \text{hpixel}) \\ & + (\text{Col factor error} \times \text{Horizontal distance from image center}) \\ & + \text{Roll error along the x axis} \end{aligned} \tag{4.32}$$

Given specifications for particular sensors, a worst case error along the x axis of the image plane could be computed using Equations (4.32) together with the algorithm presented in Equations (4.11) through (4.16).

## 4.9 Discussion

The errors with the largest effect are those along the y axis of the image plane. In particular, the most significant effect is caused by an error in vehicle to camera pitch (see Section 4.6.1). A small error in this measurement can lead to a relatively large error along the vehicle's y axis. To attempt to minimize the pitch error in the system, the vehicle to camera pitch is recalibrated at the beginning of the day, using the procedure described in Section A.2.

Most of the potential errors described in this chapter become increasingly worse as the distance from the vehicle to the projected point increases. Without exceptionally accurate calibration, the system would probably fail miserably at distances of 40 or 50 meters. STRIPE operators, however, are instructed to select points carefully, and to avoid selecting points that are so high up in the image that they can not select them accurately. When operators pick points in the 10-25 meter range, the errors described in this chapter become much less significant.

# Chapter 5 Enhancing the User Interface

## 5.1 Operator Difficulties

Early trials of STRIPE conducted at Lockheed Martin produced several unexpected complaints from operators. They reported that the system was much more difficult to use than expected, and that it took significant practice to be able to successfully navigate along the dirt roads that they used for testing. The problem did not seem to be in learning how to use the system, but with interpreting the images, and determining vehicle location.

In retrospect, this should not have been a surprise. Experience with the more traditional high-bandwidth and low-delay teleoperation shows that operators have a tendency to get disoriented and lost fairly easily, even with landmarks and a map of the area (see Sections 2.3.1.1 and 2.2.1.1). In STRIPE, the low frequency of images may actually aggravate the problem. Because there is a significant delay between images, operators cannot use the image flow to get a feel for vehicle motion.

Operators also seemed to have difficulty transferring the pan and tilt angle values to an understanding of camera orientation. For example, consider an on-road driving scenario. With the camera pointing straight ahead, the operator decides to make a left turn at an upcoming junction.

When the vehicle is still some distance from the junction, the images sent back from the camera pointing straight ahead provide a view of the current road as well as the junction, and the operator easily picks points to move the vehicle towards the junction. When the vehicle gets closer to the junction, the camera must be panned to the left to see a better view of the road behind the junction, and an image is digitized. A typical operator comment is, "If I pick this point in the image, how sharply will the vehicle turn?" The problem seems to be one of understanding which point in the real world corresponds to a chosen point in the image. This does not seem to be as much of a problem in higher-bandwidth teleoperation systems, probably because the more frequent display of new image data provides the operator with a better sense of the vehicle's motion relative to the camera direction.

Fixing the orientation of the camera on the vehicle does not solve the problem of choosing an appropriate path in a complex image. For example, consider the images in Figure 5.1. Both are images taken from a vehicle driving along a road with a fixed camera. In Figure 5.1 a, the path to choose to continue driving along the road is quite clear. However, Figure 5.1 b was taken at an intersection where a single image provides few clues as to where to steer.

a b

**Figure 5.1 (a) An easy road (b) A difficult intersection**

## 5.2 Developing the Operator Interface

It became clear from the early comments of operators that concentrating only on the vehicle side of the problem was a mistake. The current operator interface left much to be desired.

### 5.2.1 Fixing the easy problems

First, there were some basic problems that needed to be corrected. One major problem was the reporting of the pan and tilt angle. While there was a location for selecting the desired pan and tilt angles, there was no mechanism that reported the actual camera angles corresponding to the current image. This is a problem, both because of the delays in the system that might mean that the current image was taken with the old camera position, and because the actual pan/tilt position might not exactly match the requested position.

Next, "pause" and "reset" seemed to be unclear labels for the buttons, so these were renamed "stop vehicle" and "restart vehicle."

The updated version of the basic control window is shown in Figure 5.2, along with the new "camera angle information" window that shows the current camera position.

### 5.2.2 Considering the harder problems

Even with the updated windows, the operator complaints described in Section 5.1 were no closer to being solved. The complaints seemed to fit into three categories: "difficult" images (as in Figure 5.1b), confusion about camera angles, and disorientation.

*Difficult Images*

One could imagine annotating the images based on data gathered from other sensors on the vehicle in an attempt to give the operator a better understanding of what they were viewing. Short of doing that, there did not seem to be much that could be done at the interface side to help, other than offering the operator the opportunity to move the cameras and digitize another image.

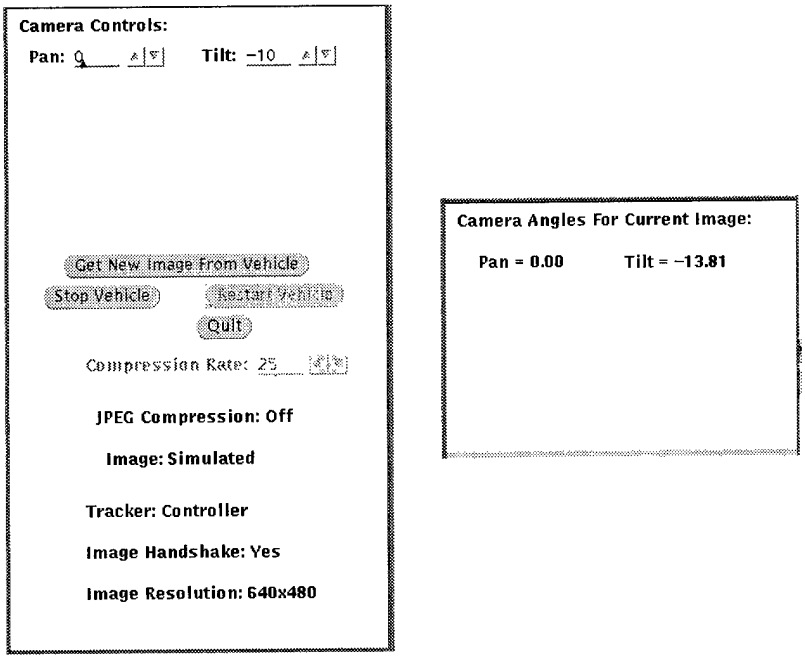



**Figure 5.2 The updated control window and angle window**

*Confusion About Camera Angles*

Of course, once the operator is allowed to move the camera, it would be nice if they were able to move it to the correct orientation. Early comments seemed to indicate that this was a more difficult problem than expected. This was a problem that might be helped by a redesign of the operator interface. Two alternative interfaces were developed.

The "dashboard interface" (Figure 5.3) attempts to show visually the camera pan by displaying where the image would appear if it were viewed through the window of a vehicle with a 100 degree field of view (about the field of view of a minivan window). The width and height of the virtual image are scaled accordingly. As the operator increases or decreases the pan value in the control window, the virtual image in the control window moves left or right as appropriate. When a new image arrives in the image window, the angle window is updated to contain the current values of the pan and tilt angles, as well as an appropriate dashboard image.

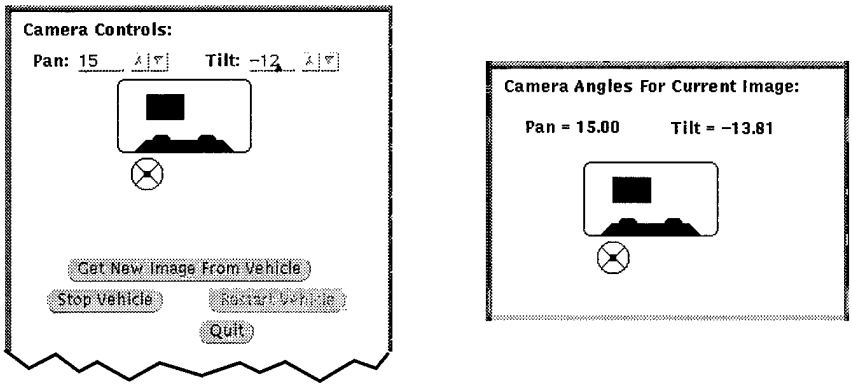



**Figure 5.3 The control and angle windows with the dashboard interface**

The dashboard interface provides a visual aid for the operator for the pan of the camera, but relies on the operator to understand tilt without a visual cue. The "compass interface" (Figure 5.4) gives the operator a visual representation of both the pan and the tilt dimensions. The left graphic on the control window is intended to represent an overhead view of the vehicle. The triangle of black represents the horizontal field of view of the camera, drawn to scale, and rotates up and down as the operator adjusts the pan. The right graphic represents a side view of the vehicle, and the corresponding vertical field of view and tilt of the camera. As the operator adjusts the tilt the field of view graphic moves appropriately. When a new image is displayed in the image window, the angle window displays both the pan and tilt values as well as the corresponding graphics.

*Disorientation*

There is some anecdotal evidence that on-line map interfaces can help reduce disorientation in high bandwidth teleoperation systems. Amai reported that when a new operator begins using a direct teleoperation system, they spend most of their time concentrating on driving the vehicle, and do not take the time to look at the dynamic map. Experienced operators will often encourage new operators to stop the vehicle and take the time to examine the map display[1]

1. Amai, Wendy S., Personal Correspondence, Sandia National Laboratories, Albuquerque, NM, November 1993.

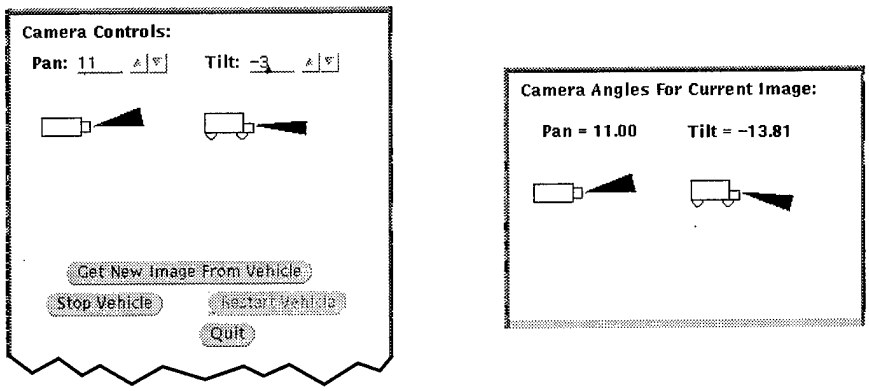



**Figure 5.4 The control and angle windows with the compass interface**

The fact that STRIPE operators have to wait between images leads to the hypothesis that a dynamic map might be even more natural for new STRIPE operators to use. A very simple map interface was developed as an aid to both new and experienced operators. This is shown in Figure 5.5.

The map shows the vehicle, and a vector pointing towards the goal. The horizontal axis of the map is parallel to the real world x-axis, and the vertical axis of the map is parallel to the real world y-axis. Every time a STRIPE image is delivered to the operator, the map is updated: the old vehicle location is faded and the new location is drawn (Figure 5.6).

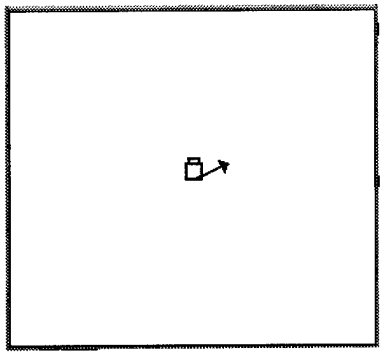


**Figure 5.5 The map interface: vehicle and direction to goal**

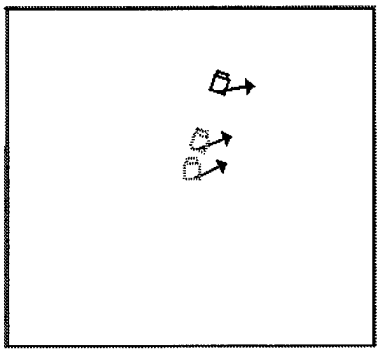


**Figure 5.6 The map interface: after a few iterations**

If the vehicle runs off the edge of the map, the map is redrawn so that the vehicle begins in its center again. When the goal is located on the visible area of the map, it is displayed as a star (Figure 5.7).

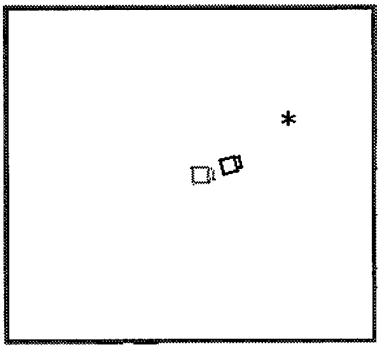


**Figure 5.7 The map interface: with goal visible.**

*An improved system?*

Intuitively, it seemed as though the changes to the interface should all be improvements. As probably the most experienced STRIPE user, I certainly believed that both the compass and map interfaces aided me in my use of STRIPE. What remained to be determined was how unbiased operators felt about the different components of the system.

# Chapter 6 Investigating the User Interface

## 6.1 Introduction

The performance of the STRIPE system is directly dependent on the performance of the operator. The real value of the system depends entirely on the ability of an operator to pick points that accurately correspond to the real world locations that the vehicle should traverse.

It was clear that information was needed about how operators make use of the system. A series of experiments was designed to test the ability of novice operators to use the STRIPE system and to compare the various interfaces.

The decision to use novice operators was based on several criteria. First, if novices performed satisfactorily, it would seem likely that expert users of the system should do better. Next, some of the potential applications of the system involved novice operators. Finally, it was more practical to use novices. The setup and run-time costs of this system were very expensive in terms of person-hours and the hours of access to the test site were limited by the owner. The site was essentially accessible to anyone who wandered by, and so any obstacles set up during the day had to be collected and brought back to the lab in the evening. Whenever the vehicle was being remotely operated, a safety driver was needed in addition to the operator trainer. In addition, the operator

environment (see Section 6.2.5) was not a particularly pleasant place to remain for an extended period of time.

# 6.2 Method

## 6.2.1 Participants

### 6.2.1.1 General Information

A total of 19 individuals participated in the experiment, which was reviewed and approved by the Carnegie Mellon University Provost's office in accordance with the requirements of Public Law 99-158 as implemented by Part 46 of Title 45 of the Code of Federal Regulations and General Assurance No. M1462.

Most participants responded to an advertisement requesting volunteers on a computer bulletin board. A few participants heard about the study through word of mouth. All participants were given a questionnaire (see Appendix C) to establish background information. Six women and thirteen men participated, with ages ranging from 20 to 52. All but one have university degrees. All participants were licensed to drive in the United States. Most were students, faculty, or staff at Carnegie Mellon University, though a few individuals associated with someone at Carnegie Mellon also participated in the study. Most of the participants have a scientific background: eight in Computer Science, three in Engineering, four in the traditional sciences, and one each in Robotics, Cognitive Science, Education & Administration, and Urban Planning. All of the participants used computers daily, and most responded that they liked computers. Most of the participants played video games less than once a week. Individuals received gift certificates for ice cream cones in exchange for taking part in the study.

Only two participants had experience with remote control devices (other than toy cars and television remotes). Operator 24, who has a degree in Robotics, had operated an adept 550 robot arm. Operator 42, with a degree in Computer Science, worked with a remotely controlled skid-steered vehicle. Both of these individuals' tests were invalidated by software and/or hardware failure (see Section 6.2.1.2).

More detailed information about the individual participants can be found in Appendix C.

#### 6.2.1.2 Invalidated Tests

Five of the participants, numbers 23, 24, 32, 42, and 43, experienced major software and/or hardware failures during their tests. Empirical data about their performance was not used from these tests, however a critical incident analysis of their tests was performed, and this verbal data has been used in this thesis.

### 6.2.2 Test Site

Tests were conducted at the Nine Mile Slag Heap, located about three miles from the Carnegie Mellon Campus. The site is closed to the public. On the majority of the site there is minimal foliage, mostly weeds and the occasional small tree, with some large trees at the edges of the site. Most of the ground is covered with slag, a by-product of the steel manufacturing industry that resembles black gravel.

28 inch bright orange traffic cones were used to define the courses. A few of the cones had reflective collars, but none of them stood out very well against the slag in the grayscale images appearing on the operator workstation. Stripes of matt blue road marking tape were added to the cones to improve their visibility in the images.

### 6.2.3 Speed Control

Because of problems with the Navlab 2's speed control system, the accelerator and brake pedals were manually operated by the safety driver. A speaker mounted over the driver's shoulder would play one sound to indicate that the driver should start to move, and another sound indicating that the brake should be applied. Safety drivers were asked to try to maintain a speed of 0.5 m/s when moving, though there was probably some variation between drivers.

### 6.2.4 Inertial Sensor Problems

All operators except 35 and 62 ran using the MIAG inertial sensor (see Section 3.5) to provide x, y, z, roll, pitch, and yaw data. Operator 35 ran in a mode where x and y were computed using dead reckoning. Z, roll, pitch, and yaw were provided by the MIAG inertial sensor. Operator 62 ran completely in dead reckoning mode. Since STRIPE essentially "resets" its position every time a new image is digitized, the results for operators 35 and 62 have minimal additional errors.

### 6.2.5 Operator Control Station

Details of the hardware used in the system can be found in Section 3.5. One important point worth repeating is the poor quality of the images on the operator's monitor. Displaying only 8 different graylevels, images were sometimes difficult to understand.

The operator control station was located in a cargo van (see Figure 6.1) A card table and two chairs were set up for the operator and observer. A mouse was attached to the laptop and used instead of the trackball. Because the wireless ethernet bridges required line of site communication, the STRIPE vehicle actually passed by the operator control station in the first task, and might have been visible through a side window on the other tasks. To prevent the operator from seeing the STRIPE vehicle, curtains were hung in front of most of the van's windows.

Power for the laptop and wireless ethernet was provided by running a power inverter through the cargo van's cigarette lighter. The cargo van worked reasonably well, but was not an ideal laboratory. While it had air conditioning, it wasn't powerful enough to really keep the entire compartment cool in the middle of summer. It was also a fairly cramped space.

The van had to be moved between the first and second tasks to maintain line-of-site contact with the STRIPE vehicle. This required the operator and observer to move to the vehicle's seats, and the operator to close his or her eyes as the vehicle was slowly driven to the site for the second and third tasks, about 100 meters away. Finally, operators had to avoid looking through the front windshield as they returned to the table.

### 6.2.6 Variables

#### 6.2.6.1 Vehicle Test Course

Each operator remotely controlled the Carnegie Mellon Navlab using the STRIPE system over 3 distinct test courses.

*Course 1: Path Following*

This course consists of an "obvious" path about 120 meters long. To ensure that the path was sufficiently obvious, traffic cones were placed on either side of the path. The path varied between about 5.5 and 6.5 meters in width, and the cones were placed about every 9-10 meters along the

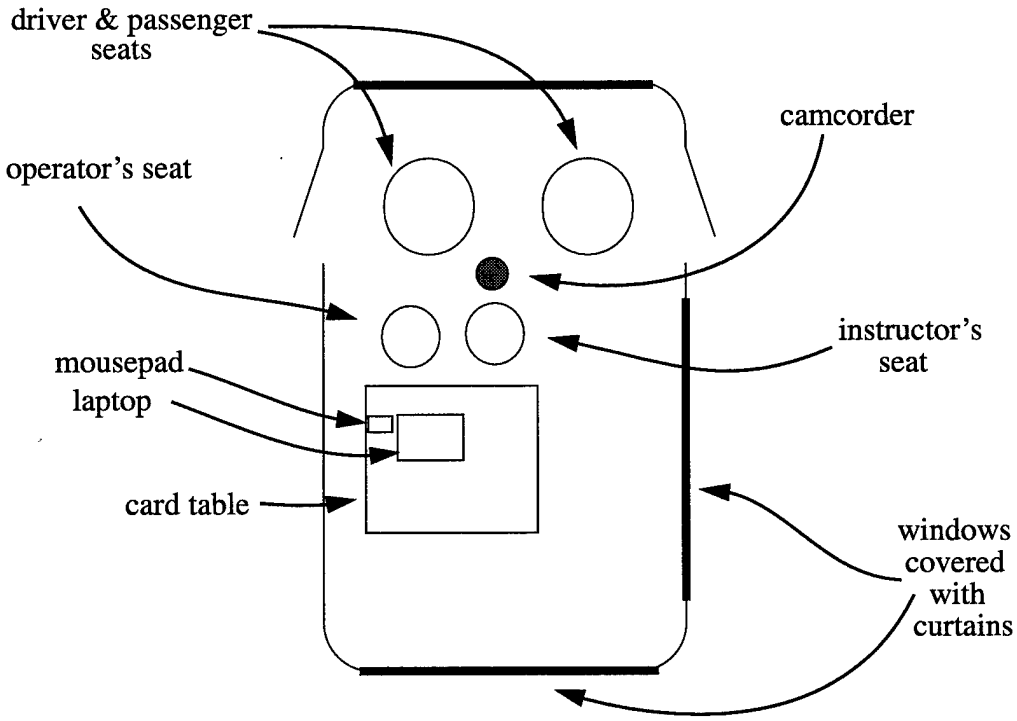



**Figure 6.1 Bird's-eye view of operator control station**

path. Between the last two cones on the path, a yellow tape was laid to designate a stop point. Figure 6.2 shows a bird's-eye view of the location of the cones for the first path, with the start position at the bottom facing the path. Figure 6.3 shows the view from the Navlab II camera at the starting position of task 1, with the cargo van highlighted.

Table 6.1 shows the approximate positions of the left and right cones for task one, along with the vehicle start position. For this data, x points East, y points North, and z points upwards towards the sky. This data was collected using the inertial sensor on the Navlab II vehicle, which has a tendency to drift somewhat over time. The data was collected by initializing at the start position and driving to each of the cones on the left hand side in order, then reinitializing at the start and driving to each of the cones on the right hand side in order. The path starts out on a paved slope, which curves round to the left and then right and becomes a more horizontal dirt and gravel road with some potholes.

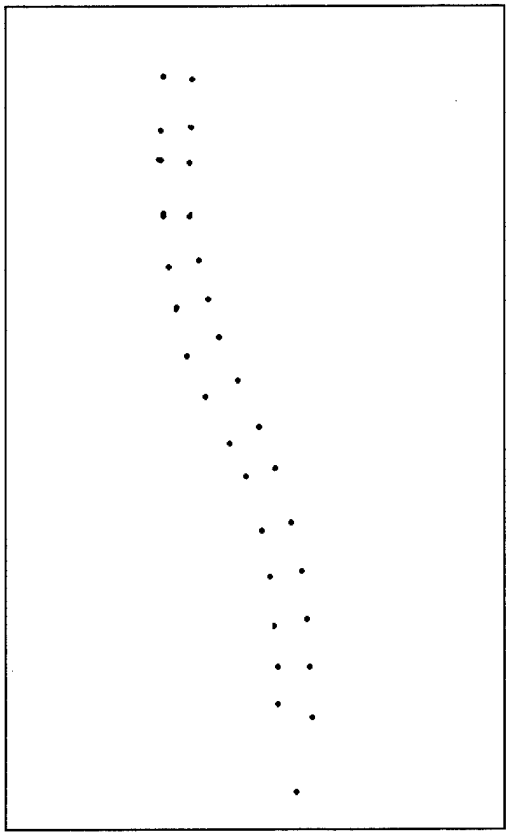


**Figure 6.2 A bird's-eye view of location of cones for path 1. The cone at the bottom of image is vehicle start position**

Operators were not walked through the course in advance. As they arrived, participants were asked to look down at the ground, and not around at the site, but most caught a glimpse of the cones on the second half of the first course before entering the cargo van.

*Course 2: Slalom*

The second course is a modified slalom. Figure 6.4 shows the approximate layout of the cones, and the dotted line indicates the intended path. The far left cone indicates the start point for the vehicle, which points between the two rows of cones. Participants are instructed to drive to the left of the single cones and to the right of the pairs of cones (see Figure 6.5) and are informed that the cones are spaced unevenly

**Figure 6.3 The view from the Navlab II camera at the start of task 1. The white circle highlights the operator cargo van, mostly obscured by the trees. This image is of a higher quality than that displayed on the operator workstation.**

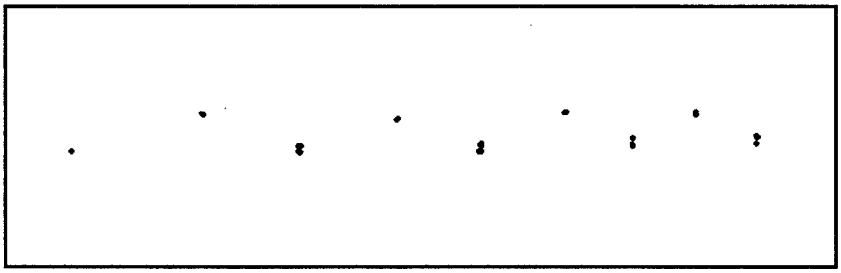


**Figure 6.4 A bird's-eye view of the layout of the cones on the second course. The task begins with the vehicle's origin on the far left cone. Operators are instructed to drive to the right of the single cones, and to the left of the paris of cones.**

The cones were offset to the left and the right to make the task possible: camera pan was limited to approximately 36 degrees from center and this meant that cones were often out of view when the camera was panned to its maximum position. Figure 6.6 shows a view of the course from the Navlab II vehicle. When seated in the vehicle, it appears at first that you could drive straight down the center of the course, between the cones, without making any steering correc-

| cone | x (meters) | y (meters) | z (meters) |
|---|---|---|---|
| start | 0.00 | 0.00 | 0.00 |
| left 1 | 8.26 | 13.28 | 0.63 |
| left 2 | 12.58 | 18.27 | 0.90 |
| left 3 | 16.89 | 23.76 | 1.20 |
| left 4 | 22.05 | 30.61 | 1.54 |
| left 5 | 26.22 | 37.12 | 1.76 |
| left 6 | 30.31 | 45.31 | 1.99 |
| left 7 | 32.26 | 51.25 | 2.13 |
| left 8 | 34.33 | 59.28 | 2.25 |
| left 9 | 36.60 | 66.54 | 2.27 |
| left 10 | 40.65 | 73.36 | 2.29 |
| left 11 | 44.47 | 79.26 | 2.36 |
| left 12 | 49.54 | 86.11 | 2.48 |
| left 13 | 55.28 | 93.15 | 2.60 |
| left 14 | 58.46 | 96.44 | 2.54 |
| left 15 | 64.97 | 102.45 | 2.60 |
| right 1 | 10.81 | 7.58 | 0.56 |
| right 2 | 16.55 | 14.30 | 0.94 |
| right 3 | 21.61 | 20.70 | 1.23 |
| right 4 | 26.64 | 27.58 | 1.56 |
| right 5 | 31.03 | 34.70 | 1.79 |
| right 6 | 34.93 | 43.04 | 2.02 |
| right 7 | 37.53 | 49.89 | 2.17 |
| right 8 | 40.24 | 57.64 | 2.28 |
| right 9 | 42.70 | 64.96 | 2.35 |
| right 10 | 45.62 | 70.98 | 2.36 |
| right 11 | 48.83 | 76.61 | 2.38 |
| right 12 | 52.86 | 82.97 | 2.55 |
| right 13 | 58.73 | 89.38 | 2.73 |
| right 14 | 62.77 | 93.44 | 2.66 |
| right 15 | 68.09 | 99.12 | 2.77 |

**Table 6.1 The approximate location of the cones in task 1. The cone labelled "start" is the vehicle start position.**

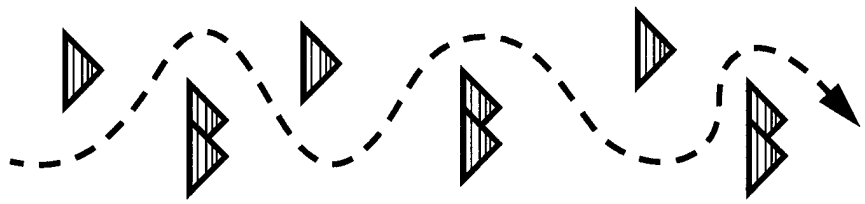


**Figure 6.5 Participants were instructed to drive to the right of the single cones and to the left of the double cones.**

tions, however this is not the case. Figure 6.7 shows the path an experienced HMMWV driver followed when instructed to drive to the left of the single cones and to the right of the pairs of cones, and to leave at least 1 foot clearance between the vehicle and the cones.

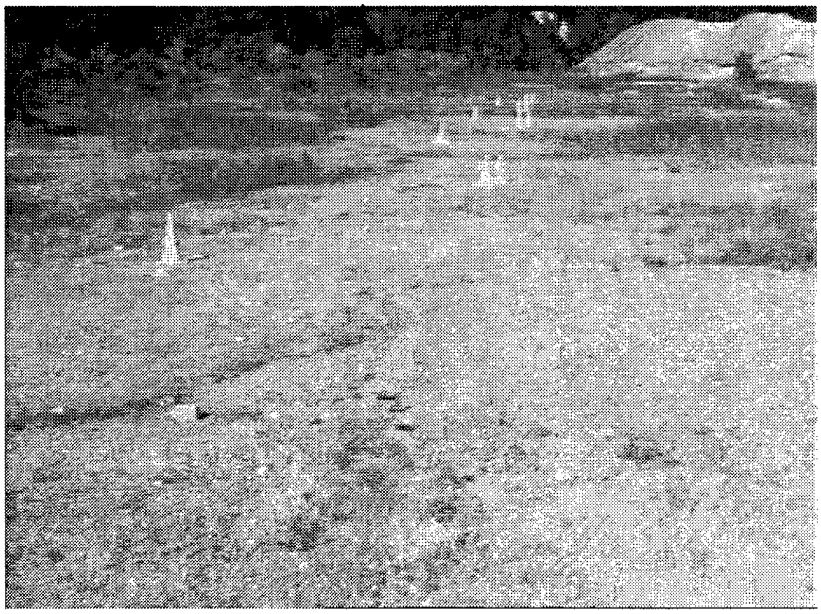


**Figure 6.6 The view from the Navlab II camera at the start of task 2. This image is of a higher quality than that displayed on the operator workstation.**

Table 6.2 shows the approximate locations of the cones for the second test course. This data was also collected using the inertial sensor on the Navlab II vehicle. To minimize the effect of sensor drift, the data was collected by initializing at the start position and driving to each of the pairs of cones, then reinitializing at the start position and driving to each of the single cones.

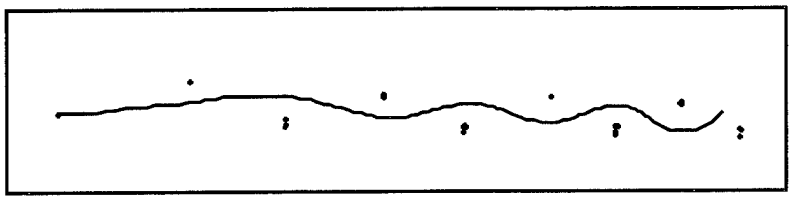


**Figure 6.7 Task 2 as driven by an experienced driver in the vehicle.**

| cone | x (meters) | y (meters) | z (meters) |
|---|---|---|---|
| start | 0.00 | 0.00 | 0.00 |
| single 1 | 3.57 | 18.38 | -0.35 |
| single 2 | 16.29 | 42.08 | 0.18 |
| single 3 | 25.89 | 63.09 | 0.41 |
| single 4 | 34.05 | 79.05 | 0.95 |
| double 1 | 13.59 | 28.42 | -0.26 |
| double 2 | 24.62 | 50.69 | 0.30 |
| double 3 | 33.25 | 69.66 | 0.77 |
| double 4 | 40.71 | 85.13 | 1.14 |

**Table 6.2 The approximate location of the cones in task2. The cone labelled "start" is the vehicle start position.**

Table 6.3 shows the distances between the cones on the slalom course. The course was intended to become more difficult as it progressed, with two sets of cones approximately 14 meters apart, then 12 meters, 10 meters, and the final set 9 meters apart. Operators did not see the task 2 course before performing the task.

| cone a | cone b | distance between a and b (meters) |
|---|---|---|
| single 1 | double 1 | 14.18 |
| double 1 | single 2 | 13.92 |
| single 2 | double 2 | 11.98 |
| double 2 | single 3 | 12.46 |
| single 3 | double 3 | 9.87 |
| double 3 | single 4 | 9.42 |
| single 4 | double 4 | 9.02 |

**Table 6.3 Distances between cones on the slalom course.**

*Course 3: Map following with obstacles*

This course makes use of the map interface (see Figures 5.5 to 5.7). Operators are instructed to drive to the goal, which is designated by a group of three cones out of the initial view of the camera. The direct route to the goal is blocked by two pairs of cones with "car dealership flags" strung between them (see Figure 6.8). Participants are instructed to avoid single cones and mounds of dirt. Figure 6.9 shows a bird's eye view of course 3. The vehicle starts at the point on the left of the diagram, pointing slightly above the horizontal.

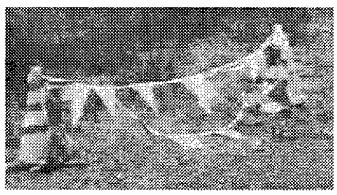


**Figure 6.8 An obstacle with "car dealership" flags.**

Table 6.4 shows the approximate locations of the cones for the final test course collected using the inertial sensor on the Navlab II vehicle. To minimize the effect of sensor drift, the data was collected by initializing at the goal position and driving to each of the cones, reinitializing three

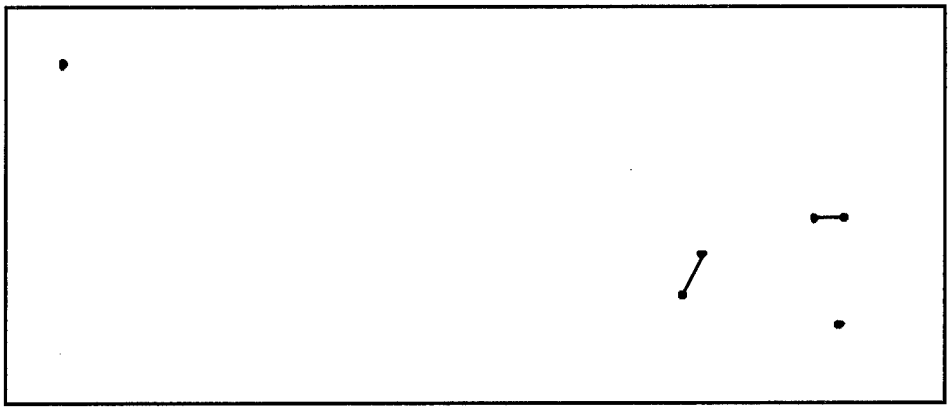


**Figure 6.9 A bird's-eye view of the layout of the cones on the final course**

| cone | x (meters) | y (meters) | z (meters) |
|---|---|---|---|
| start | 0.000 | 0.000 | 0.000 |
| barrier1 cone1 | 4.789 | -55.515 | -0.763 |
| barrier1 cone2 | 1.079 | -55.135 | -0.733 |
| barrier2 cone1 | 11.979 | -65.435 | -0.413 |
| barrier2 cone2 | 10.989 | -63.095 | -0.163 |
| goal | 3.619 | -68.295 | -0.333 |

**Table 6.4 The approximate location of the cones in task3. The cone labelled "start" is the vehicle start position.**

times at the goal position. The direct distance from start to goal ignoring the barriers is about 68 meters. Operators did not see this course before performing the task.

The initial image from the vehicle at the start of task3 was uninformative (see Figure 6.10). The purpose of the task was to see how well participants made use of the map interface to find the goal.

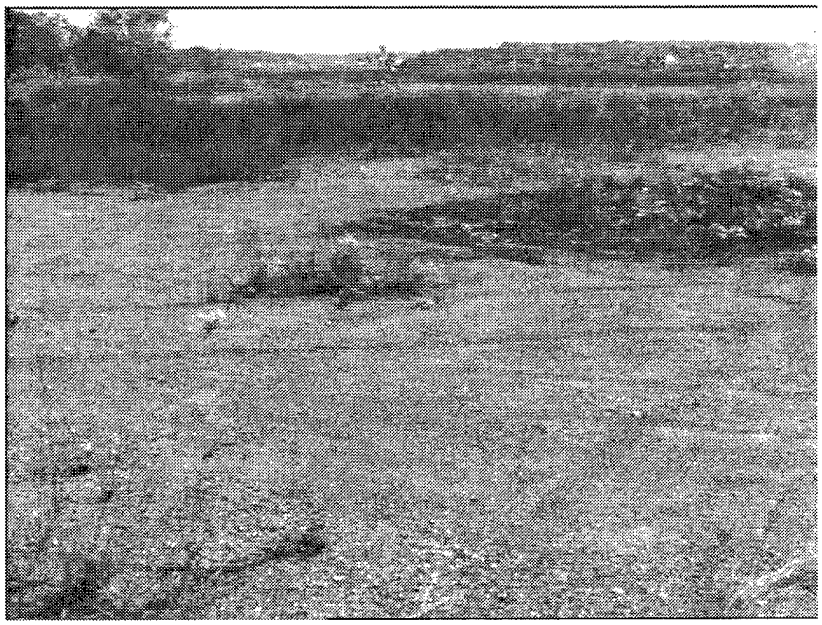


**Figure 6.10 The view from the Navlab camera at the start of task 3. No features important to the task are visible. This image is of a higher quality than that displayed on the operator workstation.**

*Variations in Tilt on Test Courses*

While one might think that the test courses were quite smooth based on the cone position data from the preceding section, this was not the case. The ground was quite uneven due to potholes and small ridges, causing the vehicle's pitch to change significantly over the course of a run.

For each test run, the minimum and maximum variation in tilt for each image was computed. Variation in tilt for an image is computed as follows: When a new image is digitized, the image_grab_tilt is reset to the vehicle tilt at the time of digitization. For each subsequent iteration of STRIPE using points picked in that image, the variation in tilt is the current vehicle tilt minus the image_grab_tilt. Table 6.5 shows the minimum and maximum variation in tilt for each of the three tasks for each of the users. Note that as was described in Section 6.2.4, user 62 ran in dead reckoning mode, and so there is no tilt data available for that run.

As was described in Sections 2.4.1 and 3.2, the strength of STRIPE over a flat-earth system is its reprojection algorithm, in which it updates its terrain model every iteration, rather than every time a new image is digitized. In the example in Sections 2.4.1 and 3.2, a change in slope of less

than three degrees caused large errors in the flat-earth system, but not in STRIPE. The data in Table 6.5 show that the tilt changed significantly over the course of many runs.

| Op # | Condition | Max 1 | Min 1 | Max 2 | Min 2 | Max 3 | Min 3 |
|---|---|---|---|---|---|---|---|
| 30 | basic | 2.67 | -2.73 | 3.18 | -2.98 | 3.05 | -1.92 |
| 44 | basic | 3.87 | -2.13 | 2.78 | -3.84 | 1.57 | -2.27 |
| 31 | no graph p/t | 3.56 | -3.27 | 2.21 | -3.24 | 2.89 | -1.57 |
| 45 | no graph p/t | 3.83 | -3.72 | 3.27 | -3.61 | 1.66 | -1.80 |
| 33 | max compress | 4.13 | -3.75 | 2.90 | -4.24 | 1.83 | -3.36 |
| 36 | max compress | 3.63 | -1.60 | 1.91 | -2.20 | 2.14 | -3.53 |
| 34 | low band | 3.33 | -1.95 | 3.18 | -3.39 | 2.29 | -2.53 |
| 35 | low band | 2.79 | -3.56 | 3.33 | -3.59 | 1.85 | -3.80 |
| 40 | windshield | 1.96 | -3.15 | 3.28 | -3.52 | 3.17 | -1.49 |
| 41 | windshield | 2.34 | -2.14 | 2.02 | -2.69 | 1.55 | -2.29 |
| 50 | wide fov | 2.57 | -3.36 | 1.77 | -3.47 | 3.73 | -3.43 |
| 51 | wide fov | 3.77 | -1.55 | 2.27 | -2.68 | 1.48 | -2.38 |
| 61 | narrow fov | 1.92 | -1.62 | 0.76 | -2.49 | 1.82 | -3.19 |
| 62 | narrow fov | - | - | - | - | - | - |

**Table 6.5 Maximum and Minimum variations in tilt for each of the three tasks measured in degrees. Variation in tilt is the negative difference between the tilt at the time an image was taken and the tilt for each iteration of stripe using that image.**

### 6.2.6.2 Operator Interfaces

*Reduced Bandwidth*

The Arlan Wireless Ethernet Bridges provided a bandwidth of approximately 500 Kbits/second. Participants who used this system used a bandwidth of approximately 125 Kbits/second The bandwidth reduction was accomplished by transmitting each byte of data across the Arlans four times. The effect of this to the operator was that the average time between a right mouse button press to send points and the new image being displayed on the operator's monitor went up to about 18.2 seconds, compared with the 12.9 seconds for the standard system (see Section 3.5 for details of the overhead included in this time).

The reduced bandwidth condition is essentially equivalent to an increased latency condition from both the operator's and the vehicle's point of view. Suppose that it normally takes i seconds to transmit an image, and p seconds to transmit the waypoints back to the vehicle. A halving of the bandwidth is equivalent to an additional latency of $\frac{p+i}{2}$ seconds. For example, consider Figure 6.11. An image starts to be digitized at time t=0, and transmission begins as soon as the digitization is complete. At time t=a, the operator with the increased latency system receives the image and picks points. The vehicle does not receive those points and begin to act on them until time t=c. The operator with the lower bandwidth system does receive the image at a later time t=b, however the vehicle still does not receive the points and begin to act on them until time t=c. Both operators picked points on an image digitized at the same time, and the vehicle received their points at the same time. Furthermore, both operators will notice a delay of c seconds between images.

*Field of view*

Glumm [12] demonstrated that for certain high-bandwidth teleoperation tasks, different fields of view were preferred. It was not clear that the results of the high-bandwidth trials would hold in the low-bandwidth and/or high-delay case, and so three different lenses were used for the STRIPE tests.

The standard system uses a camera with an 8 mm lens. A wide field of view 3.6 mm lens and a narrower field of view variable lens that was set to approximately 12 mm were also tested. The

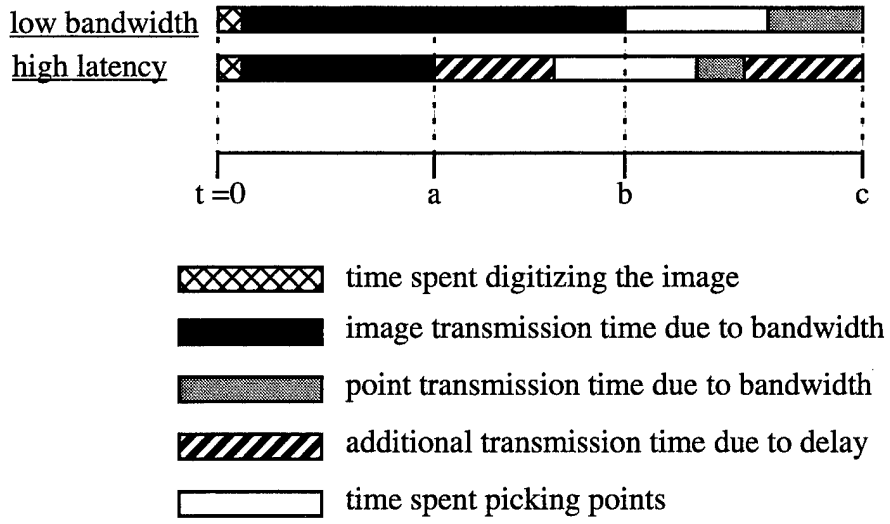



**Figure 6.11 Apparent equivalence of bandwidth and latency from the operator's point of view.**

standard camera is sealed in a weatherproof case. Instead of opening up the case to replace the lens, a second Sony XC-75 was mounted on the pan tilt next to the first, about 10 cm above and 10 cm to the left of the standard one. Neither of the lenses contained automatic iris control, and so at times the images were much brighter or much darker than the ideal. Even the sun emerging from behind the clouds could make a once perfect image nearly unviewable. Figure 6.12 shows an image taken with each of the lenses used to demonstrate the variation in field of view. These pictures also demonstrate the construction that proceeded around the test site. As the experiments progressed the sides of the roads were levelled and telephone poles were sunk into the ground. Fortunately our test areas themselves were left untouched.

As can be seen in Figure 6.13, the 3.6 mm lens introduced some significant radial distortion which could introduce serious errors into the predicted location of points chosen in the image. Rather than rectify the image, operators were presented with the distorted image, as they had been in Glumm's work [12].

To avoid the introduction of serious errors, operators were shown Figure 6.13, and told that picking points in the "warped" area would produce poor results. They were given Figure 6.14 and told that the gray areas were the best in which to pick points.

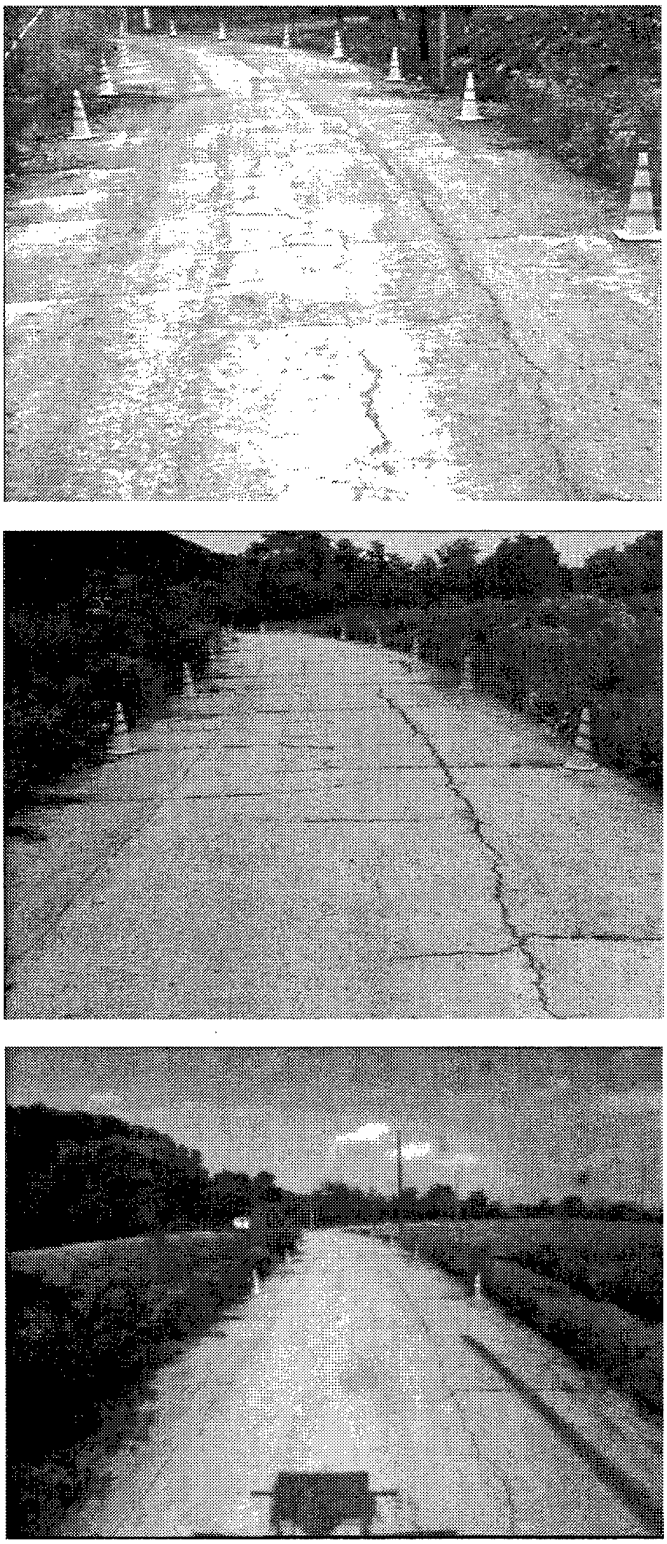



**Figure 6.12 A comparison of images from the same location taken with three different lenses. These images are of a higher quality than that displayed on the operator workstation**

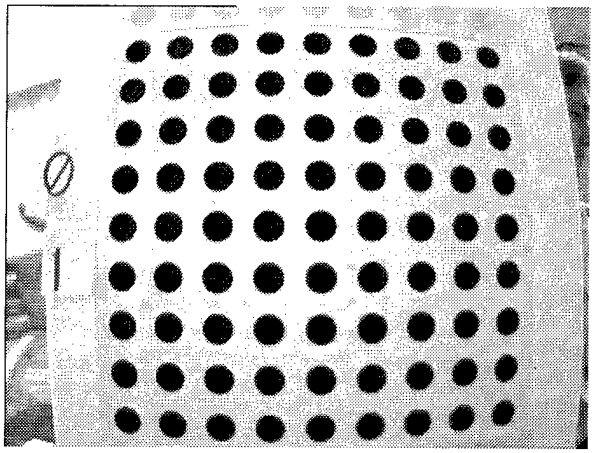


**Figure 6.13 A calibration grid as viewed through the 3.6 mm lens**

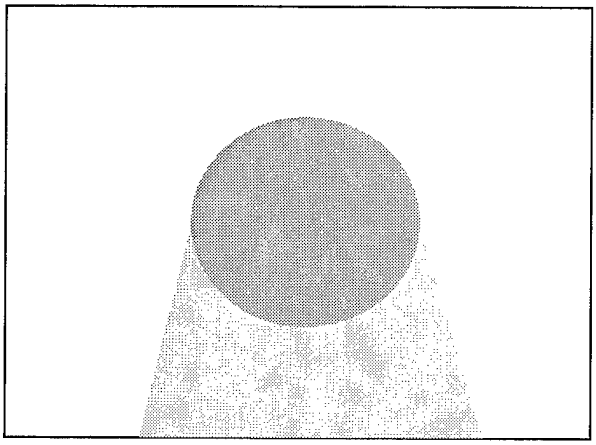


**Figure 6.14 The regions of the image that operators who used the 3.6 mm lens were instructed to use**

*Resolution vs. Image Frequency*

Under low bandwidth conditions there is a trade-off between image resolution and compressed image size. The standard system used a jpeg quality of 75 (larger values indicate more detailed images), producing images between about 40 and 60 KBytes in size. The reduced resolution system used a jpeg quality of 35, producing images between about 20 and 33 KBytes in size. Despite the significant reduction in data, the resulting images were not that much worse when

viewed on the operator workstation monitor (see Figure 6.15). The images did appear at the operator workstation slightly more quickly, due to their reduced size, on average taking 12.6 seconds to arrive, compared with the standard 12.9 seconds.

*Graphical Pan Tilt Interface*

Three different interfaces were tried out for the control of the pan/tilt adjustment on the graphical user interface. Most of the systems used the "compass interface" (Figure 5.4). Two users were tested using the dashboard interface (Figure 5.3), and two users were tested without any sort of graphical representation of pan or tilt.

#### 6.2.6.3 Summary of Conditions

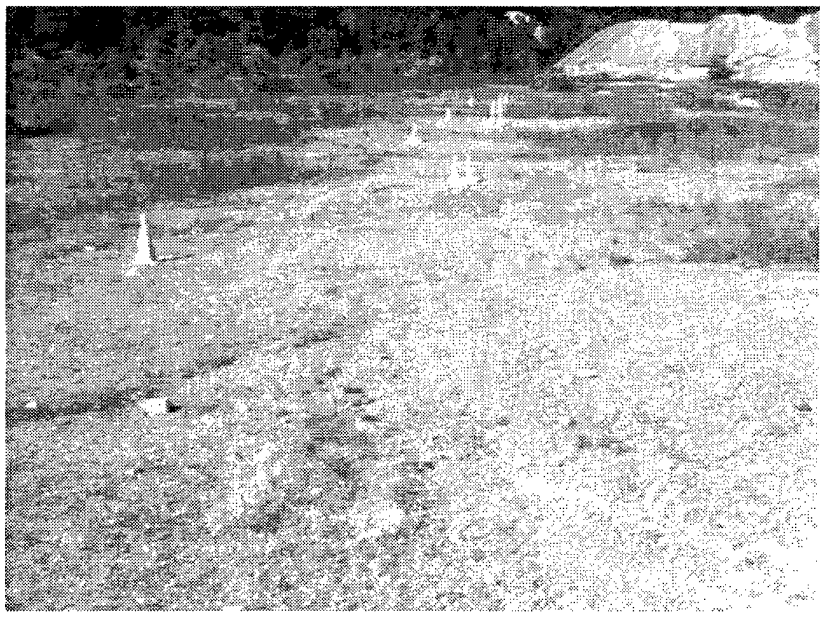
The conditions chosen for testing are summarized in table 6.6. The standard setup, called "basic", was chosen because it was the system that I, the most experienced STRIPE user, preferred to use. It has the highest bandwidth possible, thus reducing operator boredom somewhat. The medium field of view was chosen because that was the output of the standard, rainproof, Navlab 2 cameras. Minimum compression was chosen, despite the added transmission time, because it seemed to provide slightly sharper images. The compass pan/tilt interface seemed to me to be the most straightforward one to use.

### 6.2.7 Procedure

Operators were driven out to the slag heap and asked to avoid looking at the site. They were then seated at the table in the cargo van and trained to use the basic system using prerecorded image data. The training scripts used for all of the different conditions can be found in Appendix D.

A safety driver was present in Navlab 2 at all times, and had instructions to manually stop the vehicle if it was approaching a dangerous condition, or if an operator ran into a cone during a test. If the safety driver had to stop the vehicle, the operator was informed. The safety driver stopped the vehicle once because it was beginning to approach the edge of the site (operator 50's third run, see Section 6.3.3.1 for a description), and several times as operators ran into cones in the tasks (see Sections 6.3.1, 6.3.2, and 6.3.3).

Quality = 35



Quality = 75

**Figure 6.15 A significantly reduced resolution image, and a typical image, as seen on the operator workstation monitor.**

| Condition Name | Bandwidth | Lens | JPEG Quality | Graphical Pan/Tilt Interface | Participant numbers |
|---|---|---|---|---|---|
| Basic | 500 Kbps | 8 mm | 75 | Compass | 23[+a], 30, 42[+], 44 |
| Low Bandwidth | 125 Kbps | * | * | * | 32[+], 34, 35 |
| Narrow FOV | * | 12 mm | * | * | 61, 62 |
| Wide FOV | * | 3.6 mm | * | * | 50, 51 |
| Maximum Compression | * | * | 35 | * | 33, 36 |
| No Graphical Pan/Tilt | * | * | * | None | 24[+], 31, 43[+], 45 |
| Dashboard Pan/Tilt | * | * | * | Dashboard | 40, 41 |

a. Participant numbers with + were invalidated, see Section 6.2.1.2. Cells with * are the same as the "basic" condition.

**Table 6.6 Summary of test conditions**

#### 6.2.7.1 The Tests

After the participant had been trained in the use of their particular system on prerecorded image and had correctly performed certain practice tasks they began to use the real system.

*Course 1: Path Following*

The first task was intentionally straightforward, and designed so that an inexperienced operator would get a feel for the system, but probably would not have to make use of the pan/tilt mechanism. Participants were instructed to follow the path designated by the cones until they reached the line across the end of the path, and then to notify the test administrator that they had accomplished the task.

If an operator ran into a cone, the test was stopped, the Navlab was manually driven back to the start position, and the operator was allowed to repeat the test. The limited foliage at the test site still managed to interfere with our line-of-site communication. As the vehicle approached the last few cones on the first task, the data-rate would sometimes, but not always, slow to a trickle or halt altogether. If several minutes passed and a new image did not arrive, the Navlab was manu-

ally reversed so that the current image being transmitted could make it to the operator. The operator was allowed to pick points in that image, if they felt the task was not yet complete, and then the test was terminated. After they had completed the task, operators were allowed to briefly look at the task 1 course through the windshield of the cargo van before moving to the base location for the second and third tasks.

*Course 2: Slalom*

The second task was designed to force participants to make use of the pan/tilt mechanism. On this course, operators were instructed to drive to the right of the single cones and to the left of the pairs of cones, until they had passed all of the cones, and then to notify the test administrator that they had accomplished the task.

If an operator hit a cone, or went past the wrong side of a cone, the Navlab 2 vehicle was manually stopped by the safety driver, the operator was informed of this fact, and the task was concluded. Because of the proximity of the last two courses, operators were not permitted to look at the second test course until the third task had been completed.

*Course 3: Map Following With Obstacles*

The purpose of the third task was to determine how well novice operators made use of the map interface. They were told that the map that they would be using was called an "aeromap" and that it would provide them with a bird's eye view of the relative movements of the remote vehicle. If the goal was visible in the current section of the map, it would be designated by a star, otherwise an arrow indicating the direction to the goal would be present. Figure 6.16 contains the sample maps shown to the operators to help explain the interface.

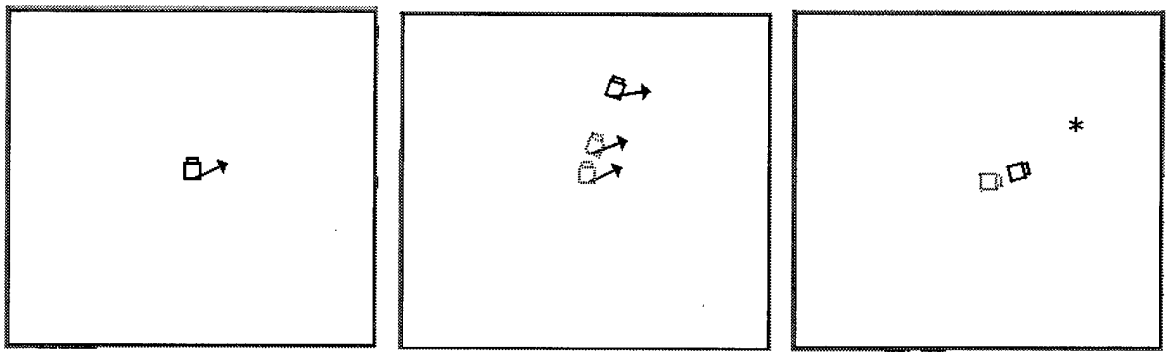


**Figure 6.16 The sample maps shown to the operators before they started task 3.**

Operators were told to move the Navlab vehicle to the goal, a trio of cones next to one another, in as direct a route as possible, while avoiding single cones and mounds of dirt. When they arrived at the goal, operators were to notify the test administrator.

## 6.3 Results

### 6.3.1 Course 1: Path Following

#### 6.3.1.1 Operator Performance

Table 6.7 compares the performance of the 14 participants in the study for whom data was evaluated (see Section 6.2.1.2). The problems with the wireless ethernet due to loss of line of sight towards the end of the course meant that the times that individuals took to complete the course were sometimes very inflated. In order to compare the performance of different individuals, the times in table 6.7 have been computed by considering the time elapsed between the start of the task and the point at which the vehicle would have crossed an imaginary line between the 4th pair of cones from the end of the run, where the line of sight was probably not a problem. The mean and standard deviation at the bottom of the table are provided to give readers a bit more feel for the spread of the data. Only one operator, number 62, hit a cone and had to repeat the course.

There is a striking difference of over 12 minutes between the fastest operator, number 36, and the slowest, number 40. What did 36[1] and 40 do differently? 36's images came slightly more quickly than 40's since they were more compressed, but even 34 and 35, the operators running under minimum bandwidth conditions, did much better than 40. 36 and 40 both travelled approximately the same distance, as did everyone in the task. By the nature of its design, one operator could not travel much further than another on the task 1 course if they stayed within the bounds of the course. 36 and 40 also picked about the same number of points in an image. There are two big differences between 36 and 40 that stand out from the data in table 6.7: first, 40 used the adjust pan control significantly more than anyone else in the study, 13 times, whereas 36 did not use it at all. Second, 40 used 33 images, almost twice the 18 that 36 used.

1. The notation xx will be used as shorthand for participant number xx throughout the rest of this chapter.

| Task 1 Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| User # | Condition | Retries | Distance travelled (meters) | Total Time (secs) | # images sent | # times points sent | # adjust pan | # adjust tilt |
| 30 | basic | 0 | 98.96 | 353 | 23 | 21 | 0 | 1 |
| 44 | basic | 0[a] | 99.33 | 547 | 23 | 21 | 2 | 0 |
| 31 | no graph p/t | 0 | 97.35 | 539 | 29 | 25 | 0 | 2 |
| 45 | no graph p/t | 0 | 99.20 | 268 | 15 | 14 | 0 | 0 |
| 33 | max compress | 0 | 99.73 | 360 | 19 | 17 | 0 | 1 |
| 36 | max compress | 0 | 99.31 | 261 | 18 | 17 | 0 | 0 |
| 34 | low band | 0 | 99.60 | 278 | 16 | 15 | 0 | 0 |
| 35 | low band | 0 | 99.47 | 356 | 20 | 18 | 0 | 0 |
| 40 | windshield | 0 | 99.68 | 995 | 33 | 20 | 13 | 0 |
| 41 | windshield | 0 | 99.05 | 721 | 30 | 27 | 0 | 0 |
| 50 | wide fov | 0 | 97.76 | 336[b] | 36 | 35 | 0 | 0 |
| 51 | wide fov | 0 | 99.74 | 627 | 31 | 27 | 2 | 0 |
| 61 | narrow fov | 0 | 98.46 | 359 | 17 | 13 | 2 | 1 |
| 62 | narrow fov | 1 | 98.05[c] | 346 | 17 | 16 | 0 | 0 |
| | | | 98.67 | 492 | 18 | 13 | 6 | 0 |
| **mean[d]** | | | **99.04** | **463.71** | **23.43** | **20.21** | **1.79** | **0.36** |
| **std dev** | | | **0.75** | **208.35** | **7.01** | **6.39** | **3.64** | **0.63** |

**Table 6.7 Summary of experimental results for task 1. "Retries" is the number of times an operator had to repeat the course. "# adjust pan" and "# adjust tilt" were the number of times an image was digitized with a change in pan or tilt from the previous image. The operators with the fastest time (36) and the slowest time (40) are highlighted.**

a. Operator 44 started this task and about a minute after the controller failed, so the test was restarted.

b. This time was artificially large because at one point the Navlab 2 safety driver did not hear the signal to start the vehicle.

c. Because user number 62 did not have any MIAG position data, the actual point when he passed the 4th cone from the end was not exact. In these runs the clock was stopped when he had travelled just under 99.04 m, the mean distance travelled over the other 13 runs. Obviously this data was not used in the computation of mean and standard deviation for this column[

d. User 62's successful second run was used in these calculations, 62's first run was not included.

These two issues are related. 40 got 12 new images without picking points in the current one, 36 never did (recall that operators never pick points in the last image that they see, so they always will have at least one more image sent to them than the number of times they sent points). Also note that 50 got more images than 40 and took significantly less time. Of the 12 new images 40 requested without picking points in the current one, 9 of them were in order to make a pan request.

Why was 40 panning the camera so much more than anyone else? To understand what was happening, it helps to look at the images that 36 and 40 saw, and the points that they picked in those images. Figures 6.17, 6.18, and 6.19 contain the complete sequence of images for the two users. Even though 36's images had maximum compression, 40's do not contain much more information that is helpful. The cones stand out fairly well in both sets of images, perhaps the stripes on the cones are not quite as apparent in some of 36's as they are in 40's, but image quality probably didn't hinder 36.

There are a few reasons why 36 probably did much better than 40. First, it looks from the images as though 40 got off on the wrong foot. Compare the first image that 40 got and the points that were picked with 36's first image and points picked. It looks as though in image 1 that 40 picked points a bit to the right of the center of the road, while 36 picked points a bit more centrally. 40's second image is almost identical to the first, and 40 picked points that seem to be more in the center of the road in that frame, but for reasons that are explained in Section 3.2, 40 was already past the last point that was picked in the second image, and so those points had no effect. When 40 received the third image the vehicle was pointing a bit to the right because of the points chosen in the first image. At this time 40 decided not to pick any points and panned the camera to the left to see the center of the road better. The fourth image shows that 40 panned 20 degrees, which was too far to the left, perhaps the dashboard interface that 40 used gave the impression that the camera had to be panned more than it actually did. Regardless of the reason for the overshoot, 40 corrected the pan to 10 degrees which was more appropriate, and finally, after having rejected two images, 40 picked some points again in image 5. Image 6 was digitized after the Navlab had moved only about a meter and a half (see Section 3.2 for an explanation) and 40 picked similar points in this image. When image 7 had arrived, 40 was more central on the road, but the camera was still panned to the left by 10 degrees and the right hand side of the path near

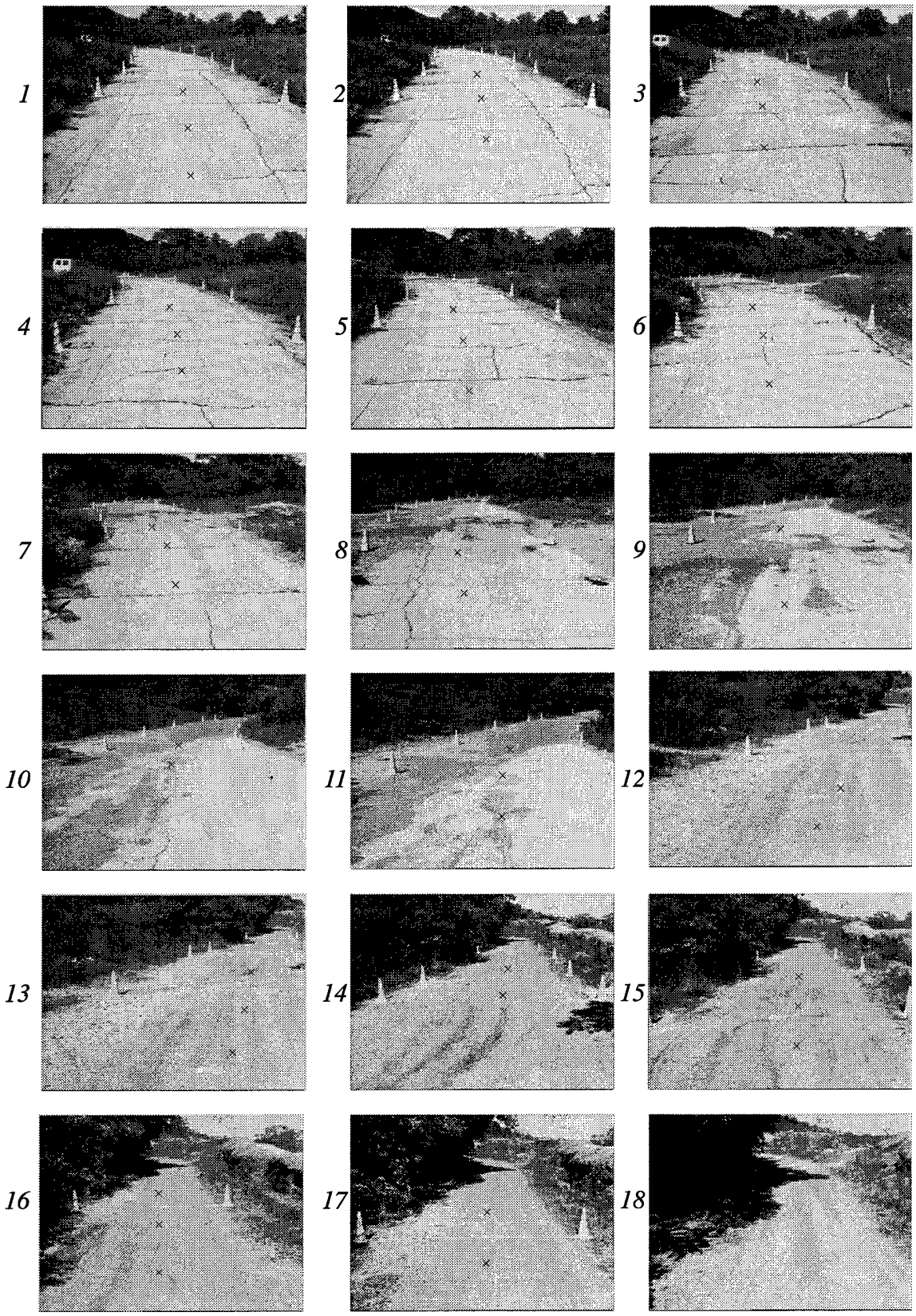



**Figure 6.17 All 18 images for The fastest run on the first course: user 36. pan = 0, tilt = -12 for all.**

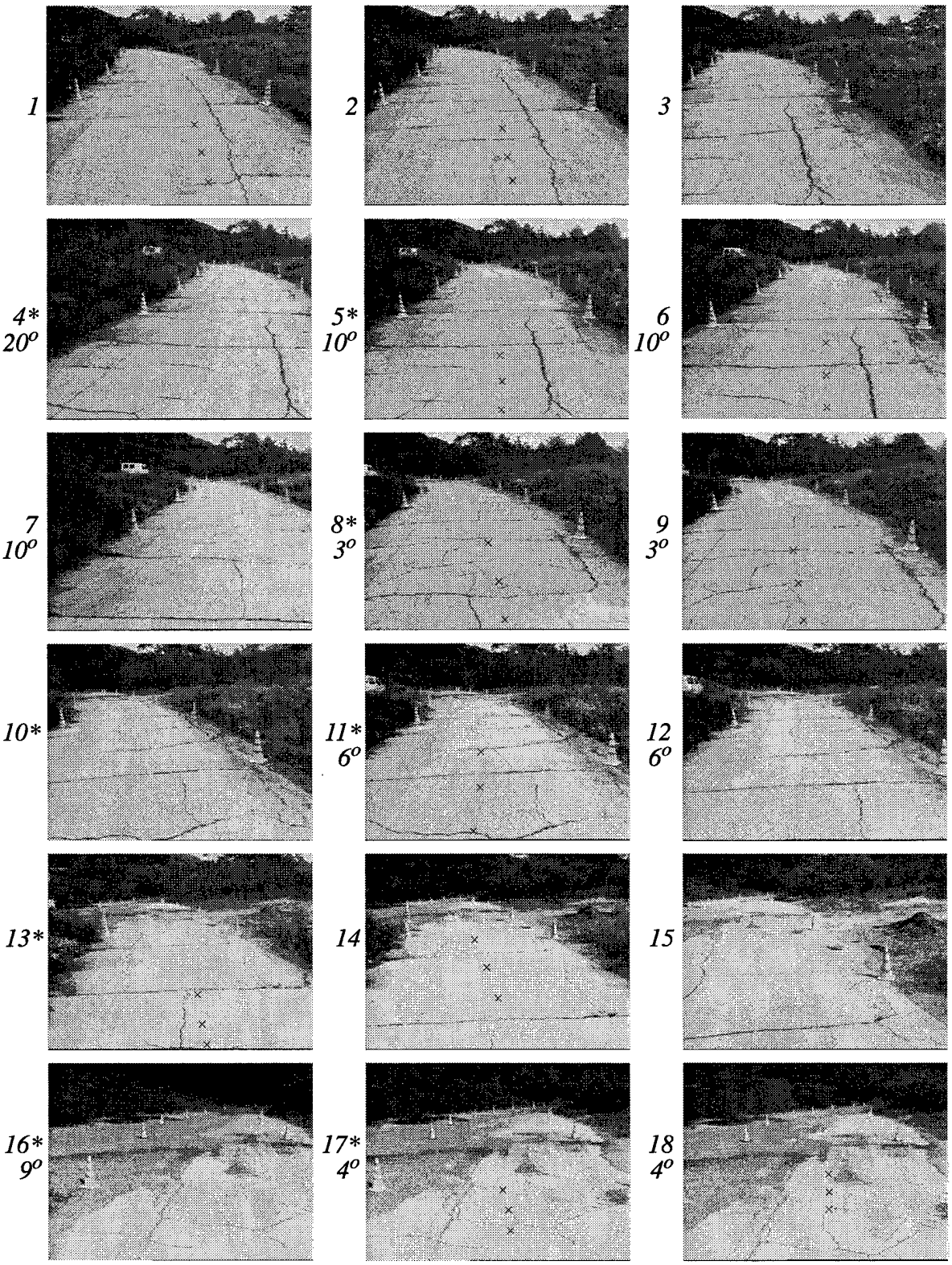



**Figure 6.18 The first 18 images and points picked for the slowest run on the first course: user 40. tilt = -12 for all, pan = 0 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans the camera to the left.**

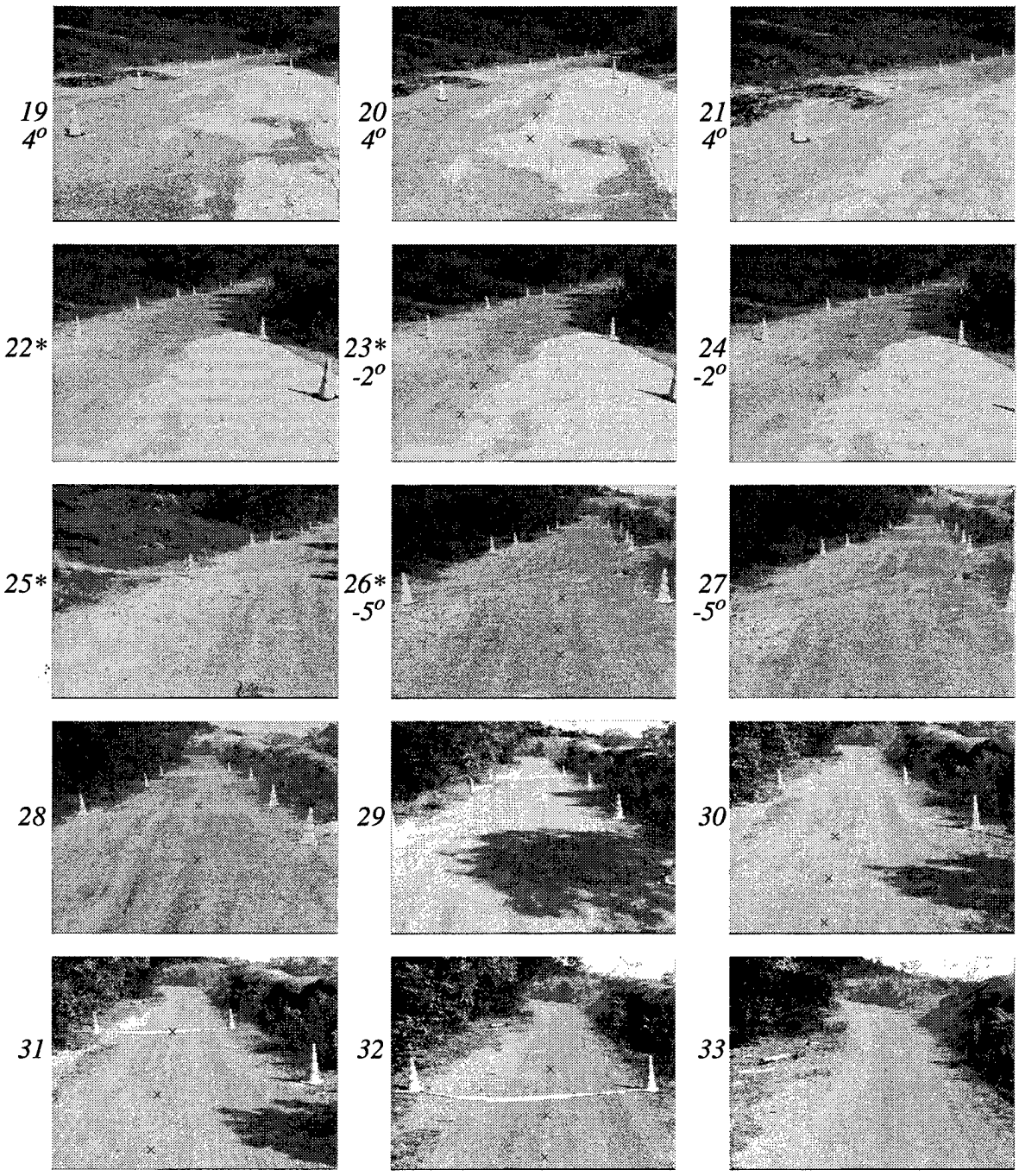



**Figure 6.19 The final 15 images and points picked for the slowest run on the first course: user 40. tilt = -12 for all, pan = 0 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans the camera to the left.**

the vehicle was not visible. So no points were picked in image 7, and the camera was panned back to the center to image 8 to pick more points. When image 8 arrived, 40 was already about 34 meters behind the location that 36 had been at when 36's eighth image appeared.

Another factor that probably had some influence in the drastic difference between the times for 36 and 40 was the difference between the locations in each image where the operators picked points. In any given image, points that are higher up in the image generally correspond to places further away in the real world. The top left corner of an image is the origin of the images coordinate system. The column number, "col" increases to the right, up to a maximum value of 639, and the row number, "row", increases down to a maximum of 479. If two points with different rows are compared, the point with the *lower* row coordinate corresponds to the point that is *higher* up in the image. Table 6.7 displays. some data about where operators picked points in their images. For each set of points picked, the maximum row value (indicating the lowest point picked in the image) and the minimum row value (the highest point picked in the image) were recorded. The "Mean Maximum Row" is the mean of all the maximum row values recorded. The "Global Maximum Row" is the largest of the maximum row values recorded. The minimums are similar.

Looking at the data for 36 and 40, we see that while they picked nearly the same number of points in each image, 36 generally picked points further out than 40. The mean minimum row value for 36 was almost 69 pixels higher up in the image than that of 40. By the time 40 sent the points from image 2 back to the Navlab, the vehicle had run out of points from image 1 and stopped moving. It had, in fact, already driven past the points chosen in image 2 (see Section 3.2 for an explanation) and so did not make any more progress, until it got the points from image 5. 36, on the other hand, picked a long path in image 1, and an even longer path in image 2, and the vehicle did not run out of points and stop until shortly after image 3 had been sent.

Notice also that the operator who had the second shortest time, 45, also had an average minimum row that was fairly small, i.e. 45 was picking points high up the image. In contrast, 41, the operator with the second longest time, had an average minimum row that was relatively large.

| Task 1 Point Picking Summary | | | | | | | |
|---|---|---|---|---|---|---|---|
| User # | Condition | Mean # points sent | Mean Minimum Row | Global Minimum Row | Mean Maximum Row | Global Maximum Row | Total Time (secs) |
| 30 | basic | 3.90 | 205.43 | 179 | 429.90 | 452 | 353 |
| 44 | basic | 2.86 | 227.29 | 185 | 415.38 | 446 | 547 |
| 31 | no graph p/t | 2.96 | 277.40 | 229 | 410.92 | 442 | 539 |
| 45 | no graph p/t | 4.00 | 151.93 | 112 | 384.07 | 420 | 268 |
| 33 | max compress | 4.06 | 182.47 | 139 | 454.12 | 470 | 360 |
| 36 | max compress | 2.88 | 193.71 | 164 | 357.29 | 409 | 261 |
| 34 | low band | 2.87 | 146.87 | 115 | 276.80 | 414 | 278 |
| 35 | low band | 2.33 | 204.56 | 140 | 351.45 | 385 | 356 |
| 40 | windshield | 3.00 | 262.45 | 167 | 408.80 | 464 | 995 |
| 41 | windshield | 4.19 | 296.78 | 124 | 459.74 | 479 | 721 |
| 50 | wide fov | 3.06 | 296.91 | 234 | 441.63 | 466 | 336[a] |
| 51 | wide fov | 2.00 | 267.48 | 215 | 410.44 | 449 | 627 |
| 61 | narrow fov | 2.62 | 205.54 | 104 | 374.08 | 415 | 359 |
| 62 | narrow fov | 3.94 | 210.88 | 120 | 400.00 | 450 | 346 |
| | | 2.69 | 139.85 | 66 | 301.92 | 383 | 492 |
| **mean** | | **3.10** | **218.48** | **155.21** | **391.18** | **435.29** | **463.71** |
| **std dev** | | **0.68** | **54.31** | **49.95** | **54.41** | **31.09** | **208.35** |

**Table 6.8 Summary of additional experimental results for task 1. "Mean # points sent" is the total number of points sent divided by the number of times points were sent. "Mean Minimum Row" is the mean of the row coordinates for the highest point picked in each image. "Global Minimum Row" is the highest point picked across all images. "Mean Minimum Row" is the mean of the row coordinates for the lowest point picked in each image. "Global Maximum Row" is the lowest point picked across all images. The operators with the fastest time (36) and the slowest time (40) are highlighted**

a. This time was artificially large because at one point the Navlab 2 safety driver did not hear the signal to start the vehicle.

One more interesting thing to note about the data in Table 6.10. 50 and 51, the operators with the wide field of view lens, had among the largest mean and global minimum row values. One might expect that this is the case because operators who used the wide field of view lens were warned not to pick points in the regions of the image that were distorted. However, both 50 and 51's mean minimum row values are actually below the center of the image, at row =239, and it was made clear to both operators that the center of the image was a good place to pick points. The real explanation is probably the fact that objects in the images taken with the wide field of view lens appear much smaller than they would in those taken with the standard and narrow lenses, and all of the operators were instructed to pick only points that were clear to them were where they wanted the vehicle to drive.

One might expect to see the corresponding effect for those with a narrow field of view. Since the narrow field of view makes objects appear larger and closer, perhaps those with a narrow field of view would pick higher in the image. 61 and 62 do have the smallest global minimum rows, i.e. they picked the highest two points picked in an image. 62 also has the smallest mean minimum row.

#### 6.3.1.2 Operator's Reactions to the Task

The quotes in this chapter are approximate reconstructions from handwritten notes taken during the tests. While they may not all be verbatim, due to difficulties of note taking in real time, all accurately reflect the sense of the participants remarks.

Most of the participants found the first task to be fairly simple.

*35: Task 1 was pretty straightforward.*

*31: The first task was easy.*

41 had two opinions on the subject:

*41: This is totally fun.*

*41: Actually, this is a little dull, I'm going to pick more points.*

The was certainly the simplest task. Lots of cones defining a course with only one, easy to navigate, S-curve. It is important to emphasize, however, that all but one of the operators successfully navigated this course on the first attempt. The only experience they had with the system

before using it in this task was practice laying points down in a set sequence of prerecorded images that did not react differently to different sets of points.

### 6.3.2 Course 2: Slalom

#### 6.3.2.1 Operator Performance

Course 2 was designed to force operators to pan the camera on the Navlab vehicle, and it certainly succeeded. In course 1 less than half of the participants used the pan, and most of those who used it only used it twice. In course 2 only three participants did not use the pan.

Course two proved to be a much more difficult task than was imagined when it was designed; only one operator made it to the end of the course successfully. The difficulty of the course was aggravated by the poor quality of the images, making it even harder for an operator who had lost track of the cones to see them as they panned around looking for them. Finally, an undocumented safety feature in the vehicle's steering controller caused the vehicle to drive straight ahead when extremely sharp turns were requested. This was not discovered until the last few subjects were tested. Some operators might have performed better had this feature been disabled sooner, though as we shall see, it might have helped the operator who successfully traversed the entire course get past the last pair of cones.

Table 6.9 summarizes the data collected for the second task.

It was surprising to discover that the only operator who completed the entire course did not use pan control even once. What was special about 34's run that made it successful without even one adjustment to the camera angle? Figure 6.20 contains the 12 images for 34's run, and the points that were picked. Considering the images, it's clear that 34 almost always got images that were looking down the center of the slalom, and there was little need to pan. Was this just a coincidence?.

When operators were instructed in the use of the system, it was made clear to them that the last two points that they picked in an image would determine the orientation of the vehicle when the vehicle reached the end of a given path. It was expected that operators would use this fact in the slalom to help them position the vehicle for the next move. However, as will be discussed in

| Task 2 Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| User # | Condition | # obsta-cles com-pleted | Distance Travelled (meters) | Total Time (secs) | # images sent | # times points sent | # adjust pan | # adjust tilt |
| 30 | basic | 4 | 69.12 | 774 | 18 | 12 | 5 | 0 |
| 44[a] | basic | 4 | 69.94 | 871 | 20 | 11 | 8 | 0 |
| 31 | no graph p/t | 2 | 46.38 | 431 | 11 | 6 | 4 | 0 |
| 45 | no graph p/t | 5 | 77.29 | 326 | 13 | 9 | 0 | 0 |
| 33 | max compress | 7 | 93.35 | 757 | 19 | 15 | 6 | 0 |
| 36 | max compress | 5 | 72.06 | 386 | 12 | 9 | 2 | 0 |
| 34 | low band | 8 | 101.52 | 570 | 12 | 11 | 0 | 0 |
| 35 | low band | 5 | 78.11 | 841 | 18 | 14 | 3 | 0 |
| 40 | windshield | 5 | 77.85 | 603 | 16 | 13 | 2 | 0 |
| 41 | windshield | 2 | 43.57 | 533 | 14 | 6 | 2 | 0 |
| 50 | wide fov | 3 | 58.56 | 287 | 14 | 12 | 1 | 0 |
| 51 | wide fov | 4 | 67.16 | 403 | 13 | 12 | 0 | 0 |
| 61 | narrow fov | 5 | 75.70 | 671 | 18 | 8 | 7 | 0 |
| 62 | narrow fov | 4 | 61.77 | 766 | 15 | 5 | 8 | 0 |
| **mean** | | **4.5** | **70.88** | **587.07** | **15.21** | **10.21** | **3.43** | **0** |
| **std dev** | | **1.65** | **15.71** | **197.74** | **2.94** | **3.12** | **2.93** | **0** |

**Table 6.9 Summary of experimental results for task 2. "# obstacles completed" is the number of cones and cone-pairs that the operator successfully navigated past (single and double cones each count as "one cone" in this calculation). As soon as an operator hit a cone or went around the wrong side of a cone the task was concluded. The operator who performed the best (34) and one of the worst (41) are highlighted.**

a. Operator 44 did task 1 on one day, and tasks 2 and 3 the next day due to controller failure.

Section 6.4.1.1, STRIPE often sent the new image *before* the previous path had been completed, and this confused operators who saw an image taken from an unexpected position and orientation.

In 34's case, the first four paths designated in images 1-4 were fairly straight, and so an image that was digitized before the vehicle reached the end of it's path was not too confusing, since it

| Task 2 Point Picking Summary | | | | | | | |
|---|---|---|---|---|---|---|---|
| User # | Condition | Mean # points sent | Mean Minimum Row | Global Minimum Row | Mean Maximum Row | Global Maximum Row | Total Time (secs) |
| 30 | basic | 3.08 | 290.83 | 205 | 433.75 | 456 | 774 |
| 44 | basic | 2.81 | 292.36 | 231 | 424.18 | 458 | 871 |
| 31 | no graph p/t | 3.78 | 210.67 | 124 | 404.17 | 424 | 431 |
| 45 | no graph p/t | 7.67 | 183.44 | 136 | 383.00 | 410 | 326 |
| 33 | max compress | 4.80 | 260.20 | 166 | 456.13 | 474 | 757 |
| 36 | max compress | 2.78 | 197.00 | 124 | 322.78 | 353 | 386 |
| 34 | low band | 2.91 | 175.27 | 112 | 262.55 | 393 | 570 |
| 35 | low band | 2.21 | 257.86 | 173 | 347.64 | 438 | 841 |
| 40 | windshield | 2.92 | 245.69 | 199 | 394.00 | 435 | 603 |
| 41 | windshield | 3.67 | 301.33 | 213 | 447.50 | 479 | 533 |
| 50 | wide fov | 3.50 | 294.00 | 201 | 406.75 | 458 | 287 |
| 51 | wide fov | 2.50 | 257.83 | 222 | 417.50 | 462 | 403 |
| 61 | narrow fov | 2.63 | 196.75 | 67 | 396.00 | 440 | 671 |
| 62 | narrow fov | 3 | 234.60 | 92 | 405.00 | 479 | 766 |
| **mean** | | **3.45** | **242.70** | **161.79** | **392.93** | **439.93** | **587.07** |
| **std dev** | | **1.38** | **43.83** | **52.50** | **51.71** | **35.68** | **197.74** |

**Table 6.10 Summary of additional experimental results for task 2. See Table 6.10 for an explanation of the headings. The operator who performed the best (34) and one of the worst (41) are highlighted.**

would still produce a direct view of the cones. In images 5 and 6, 34 starts picking paths that move the vehicle off to the right. By image 7 the cones have moved to the left of the image, but all are still visible, and given the paths chosen in images 5 and 6, image 7 is a reasonable result to expect. Between images 7 and 11, 34 has a good view of the entire course (a set of double cones is indeed visible in image 9 if you squint. At image 11, our hero 34 makes what might have been a fatal mistake. Pay attention to the mound in the background and how it moves between images 10 and 11. It looks as though the cone pictured in image 11 is actually the right one in the last pair of cones, and image 11 was digitized as the vehicle swung out to the right at the beginning of the

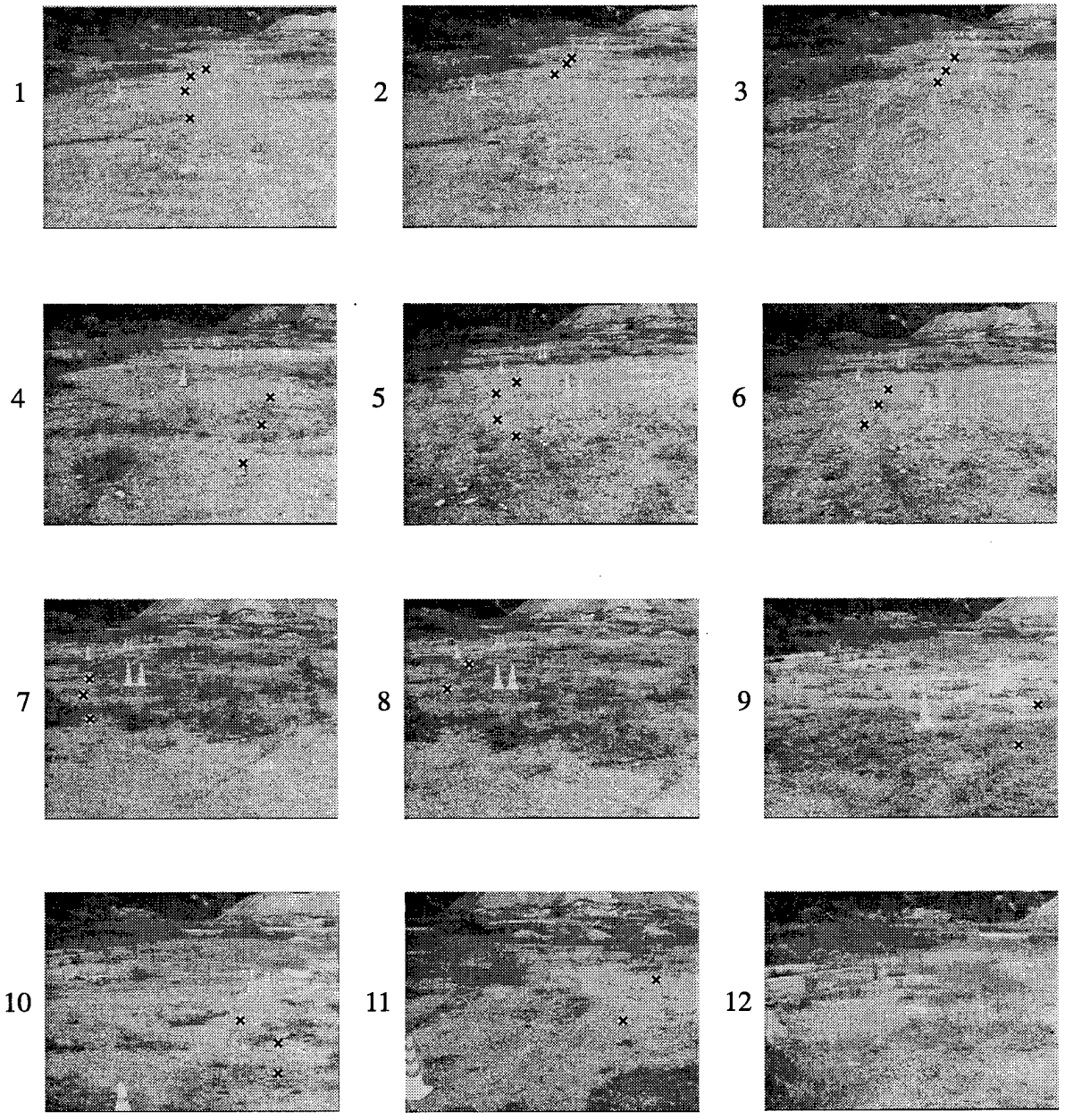



**Figure 6.20 All 12 images and points picked for operator 34, the only operator to complete the slalom course. pan = 0 and tilt = -12 for all. The x's have been widened to make them more visible to the reader.**

path in image 10. 34 picked points to the right of that cone, i.e. on the wrong side of the cones. But, by the time those points made it back to the vehicle, it had followed more of the path in image 10, and was now heading towards the correct side of the cones. At this point one of two things happened. Either the vehicle was past the beginning of the path designated in 11 and so

tried to plot a smooth course back onto it, which moved it barely past the left side of the last pair, or the undocumented safety feature in the controller took over, determined that the vehicle was trying to make too sharp a turn, and drove straight. In any event, 34 correctly traversed the course, despite the error in the last set of points chosen.

Two final points to note about the images in 34's run. First, 34 did have the smallest mean maximum row, i.e. 34's last point was, on average, much higher than anyone else's. Also, certain pairs of images, 1 & 2, 5 & 6, and 7 & 8 look very similar to each other. The reason for this will be explained in Section 6.4.1.2.

In 34's images, the first few cones in the slalom course seem quite trivial to successfully traverse, how could someone fail to get past more than two cones successfully? Figure 6.21 documents operator 41's collision with the second single cone (i.e. the third obstacle in the slalom). 41's run began uneventfully, and by image 3 the first single cone on the left was out of view. Points chosen in images 5 and 6 successfully navigate the vehicle around the second obstacle, a double cone. But the points chosen in image 9 were a poor choice. Images 8 and 9 are essentially the same, the vehicle had stopped moving before image 8 was taken, and since no points were sent and the camera had not been panned, image 9 looks identical. 41 picks points in image 9 that head the vehicle towards the cones, but if that line were to continue, it would go past the wrong side of the third obstacle. By the time image 11 was digitized, the third obstacle was very close to the bottom of the screen. Note that 41 did pan the camera to the right before image 11 was taken, but not far enough to the right to see much around the right side of that obstacle. Instead of panning the camera further and picking points around the right side of the cone, 41 tries to pick a point (actually, two points very close together) at the very bottom of the image, thinking that this might move the vehicle to the right (since the camera was panned to the right). While the logic is sound, by the time the front of the vehicle arrived at the point that 41 had designated, the vehicle was too close to the cone to be able to maneuver around it. In fact, the cone was not visible in images 13 and 14. It might have been viewable had 41 tilted the camera down further.

One other thing to note about this task: 41 often requested new images without picking points. This was a conscious decision by 41 that had a reasonable explanation that will be discussed in Section 6.4.1.1.

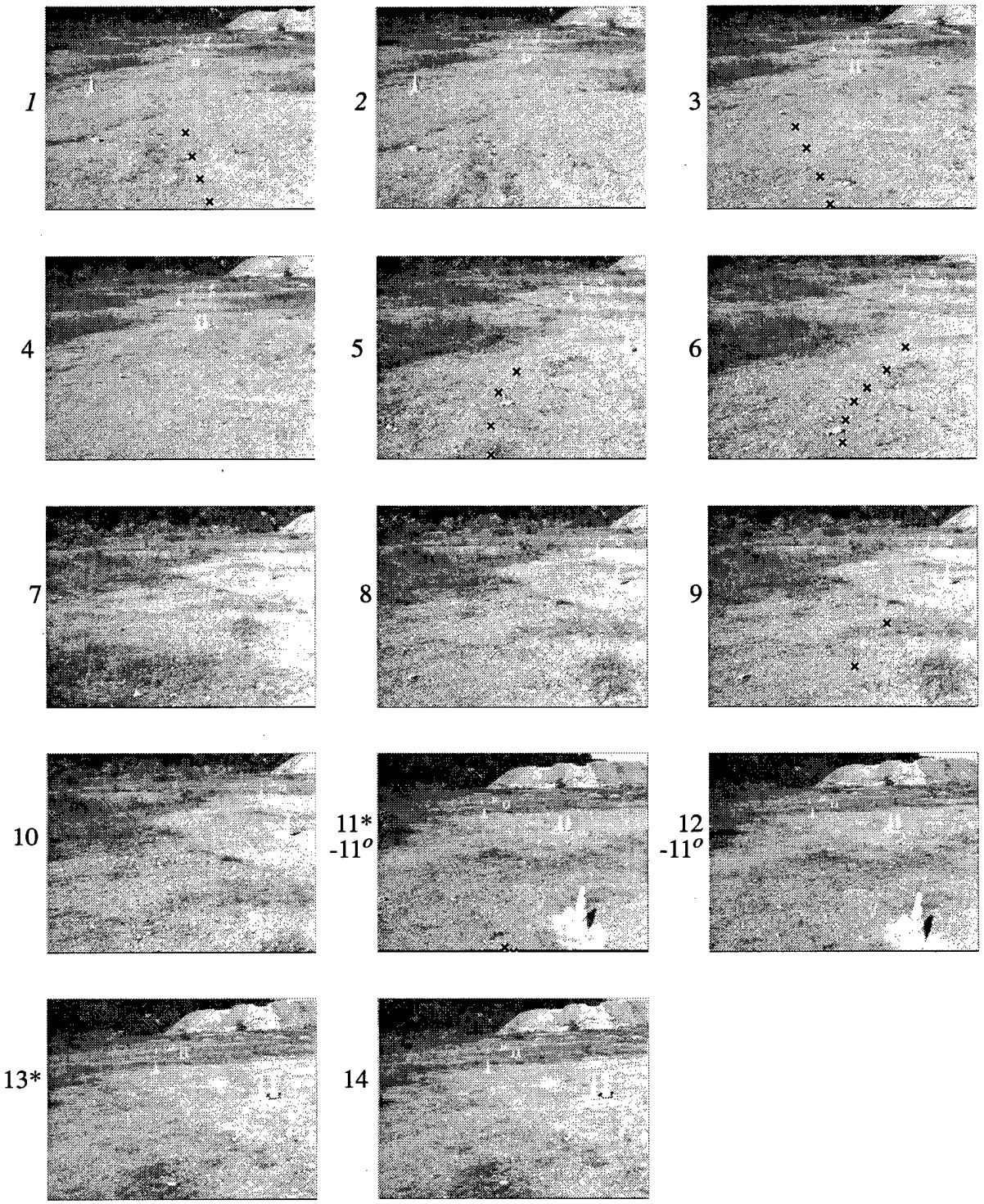



**Figure 6.21 All 14 images and points picked for operator 41, one of the two operators who had the worst performance on the slalom course. tilt = -12 for all, pan = 0 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans the camera to the left.**

#### 6.3.2.2 Operator's Reactions to the Task

Most operators found the slalom to be the most difficult of the three tasks.

*41: The slalom zigzag course was more difficult than it looked.*

*35: It was difficult to get feedback from slalom.*

It was a task in which the vehicle's orientation was changing often, and unlike in task 3, the operator did not have any feedback as to the orientation of the vehicle at any given time. The operators were also required to make difficult steering maneuvers in a very limited area.

### 6.3.3 Course 3: Map following with obstacles

#### 6.3.3.1 Operator Performance

The purpose of task 3 was to see how well participants made use of the map interface, referred to in the testing as the "aeromap". The primary metric for success in this task was distance travelled, and operators were instructed to take the most direct route to the goal, while avoiding obstacles. Tables 6.11 and 6.12 summarize the results from the third task.

Due to a misunderstanding, the safety driver oriented the vehicle for operators 30, 31, 33, and 34 incorrectly, with the vehicle at a yaw of approximately 180 degrees and pointing directly at the goal, rather than 240 degrees, and pointing off to the side. 33 and 34 both tried to turn the vehicle around by 180 degrees upon seeing the first aeromap, so they did not immediately recognize the goal (this is described further in Section 6.3.3.2). 30 and 31's first images were lost. The points that they picked headed towards the goal, but the image quality was poor enough that to the best of my knowledge these operators did not see the goal in their first image.

The most direct successful route by an operator at the correct initial orientation was taken by operator 35, who made it to the goal in just under 70 meters. Figure 6.22 shows the images and points picked by 35. The route is fairly uneventful. 35 begins by picking points off to the right (the direction of the goal), and later pans the camera a little to the right to pick more points. 35's path of approach takes the vehicle in between the pairs of "car dealership" flags to the goal. The vehicle continued moving after image 15 was taken, but 35 did not request a new image to see the next image. Notice that certain pairs of images again look similar to each other: 1 & 2, 3 & 4, 5 & 6, 7 & 8, 9 & 10, and 14 & 15. This will be explained in Section 6.4.1.2.

The longest route by far was taken by 50, one of the operators with a very wide field of view lens. 50 started out heading in the correct direction, but drove past the goal. As 50 drove towards an unsafe area of the test site, the vehicle was taken out of autonomous mode, turned around, and control was given back to 50, however the vehicle never made it to the goal. One major reason for this was the poor quality of 50's images. The sun was pointing at the camera which produced some cone-like reflections in the image, and the lack of an automatic iris on the camera produced images with very little detail. Figure 6.23 shows a few of the images that 50 saw. Looking closely

| Task 3 Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| User # | Condition | suc-cess? | Distance Travelled (meters) | Total Time (secs) | # images sent | # times points sent | # adjust pan | # adjust tilt |
| 30[a] | basic | Y | 63.73 | 668 | 17 | 14 | 2 | 3 |
| 44 | basic | Y | 72.47 | 534 | 16 | 13 | 0 | 0 |
| 31 | no graph p/t | Y | 68.21 | 710 | 21 | 10 | 10 | 0 |
| 45 | no graph p/t | Y | 72.16 | 482 | 13 | 8 | 4 | 0 |
| 33 | max compress | Y | 65.49 | 283 | 10 | 9 | 0 | 0 |
| 36 | max compress | N[b] | 56.07 | 389 | 11 | 6 | 2 | 0 |
| 34 | low band | Y | 77.63 | 296 | 9 | 8 | 0 | 0 |
| 35 | low band | Y | 69.51 | 675 | 15 | 11 | 3 | 0 |
| 40 | windshield | Y | 83.25 | 874 | 28 | 17 | 7 | 0 |
| 41 | windshield | N[c] | 80.13 | 1070 | 30 | 13 | 5 | 0 |
| 50 | wide fov | N[d] | 186.05 | 1131 | 53 | 46 | 3 | 0 |
| 51 | wide fov | Y | 72.10 | 310 | 14 | 13 | 0 | 0 |
| 61 | narrow fov | Y | 76.04 | 332 | 10 | 5 | 2 | 1 |
| 62 | narrow fov | N[e] | 72.81 | 297 | 7 | 5 | 2 | 1 |
| **mean** | | | **79.69** | **575.07** | **18.14** | **12.71** | **2.86** | **0.36** |
| **std dev** | | | **31.39** | **290.23** | **12.13** | **10.23** | **2.91** | **0.84** |

**Table 6.11 Summary of experimental results for task 3. "# cones completed" is the number of cones the operator successfully navigated around. As soon as an operator hit a cone or went around the wrong side of a cone the task was concluded. The operators who started with the correct orientation and completed the course and who travelled the shortest total distance, 35, and the longest total distance, 40 are highlighted.**

a. Operators 30, 31, 33, and 34 began at a different orientation, see text.

b. Operator 36 ran up to one set of "car dealership flags"

c. Operator 41 had to pause briefly because of controller failure, and eventually ran over one set of "car dealership flags".

d. Operator 50 had extremely poor quality images, see text.

e. Operator 62 drove towards an unsafe area and the test was stopped.

| Task 3 Point Picking Summary | | | | | | | |
|---|---|---|---|---|---|---|---|
| User # | Condition | Mean # points sent | Mean Minimum Row | Global Minimum Row | Mean Maximum Row | Global Maximum Row | Total Time (secs) |
| 30 | basic | 3.43 | 248.14 | 154 | 428.07 | 445 | 668 |
| 44 | basic | 2.77 | 271.38 | 181 | 426.85 | 466 | 534 |
| 31 | no graph p/t | 6.00 | 288.10 | 211 | 424.40 | 438 | 710 |
| 45 | no graph p/t | 3.00 | 250.75 | 185 | 388.00 | 405 | 482 |
| 33 | max compress | 2.85 | 183.78 | 117 | 449.22 | 464 | 283 |
| 36 | max compress | 3.00 | 214.00 | 174 | 372.50 | 412 | 389 |
| 34 | low band | 3.00 | 139.38 | 64 | 222.13 | 369 | 296 |
| 35 | low band | 2.00 | 256.91 | 218 | 333.73 | 403 | 675 |
| 40 | windshield | 2.24 | 309.94 | 220 | 422.65 | 454 | 874 |
| 41 | windshield | 2.85 | 395.23 | 274 | 468.31 | 479 | 1070 |
| 50 | wide fov | 3.13 | 301.3 | 250 | 439.37 | 468 | 1131 |
| 51 | wide fov | 2.54 | 254.54 | 222 | 442.15 | 459 | 310 |
| 61 | narrow fov | 2.80 | 197.40 | 117 | 396.00 | 421 | 332 |
| 62 | narrow fov | 2.60 | 182.20 | 94 | 335.20 | 479 | 297 |
| | | **3.02** | **249.50** | **177.21** | **396.33** | **440.14** | **575.07** |
| | | **0.93** | **64.79** | **61.35** | **64.44** | **33.33** | **290.23** |

**Table 6.12** **Summary of experimental results for task 3. See table 6.12 for an explanation of the headings. The operators who started with the correct orientation and completed the course and who travelled the shortest total distance, 35, and the longest total distance, 40 are highlighted.**

at the third image one can make out a set of "car dealership flags" on the left hand side of the image. Not knowing what to look for, 50 did not even notice them. 50 never made it to the goal. 50 did ask an important question about the map interface during this task:

*50: Is the truck on the aeromap to scale? If it is I can use it to judge how I'm doing.*

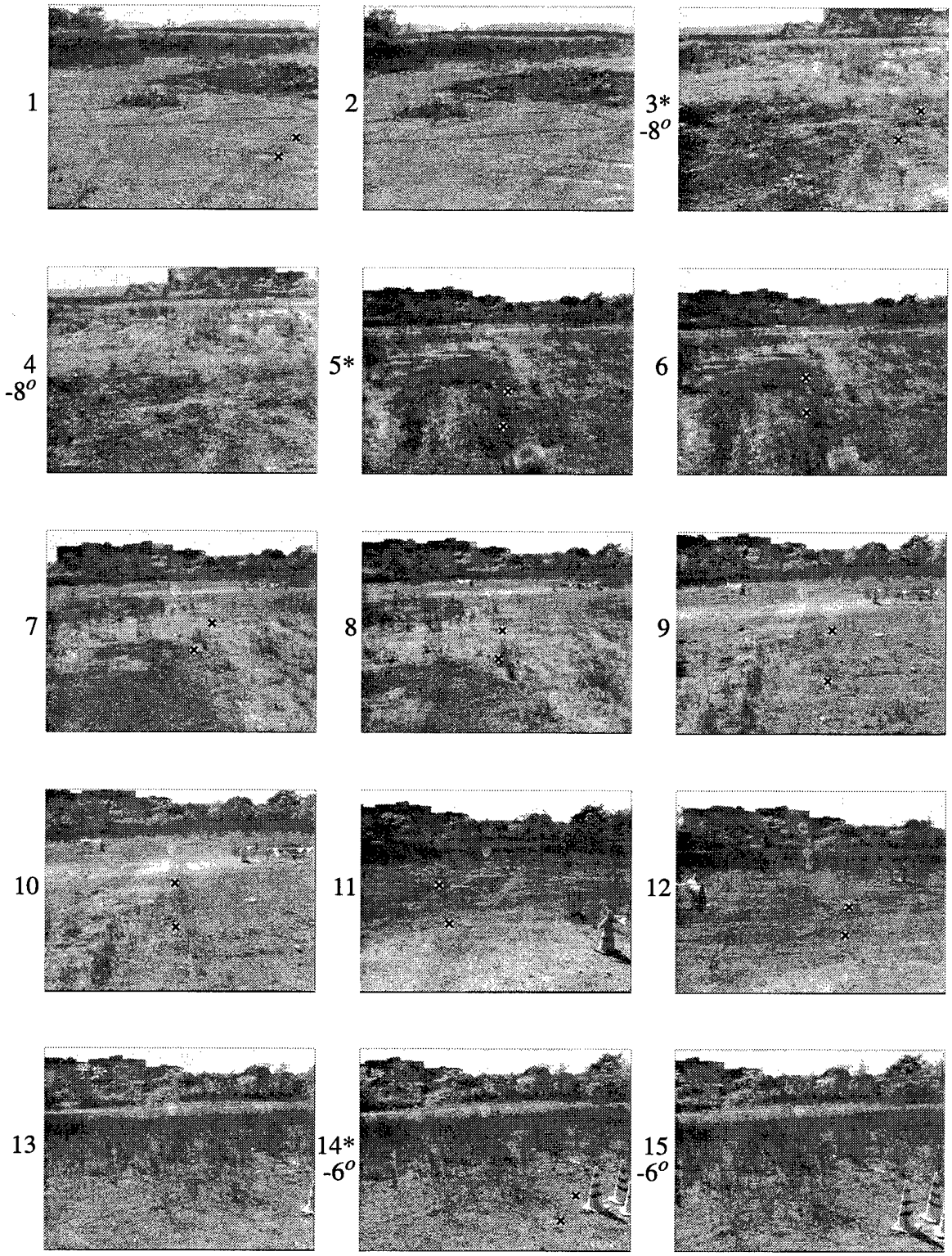
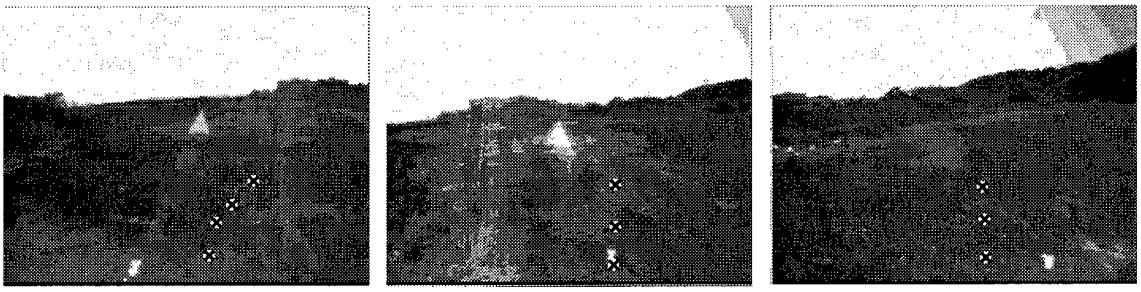



**Figure 6.22 All 15 images and points picked for operator 35 who had the shortest path in the last task. Pan = 0 and Tilt = -12 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans to the left.**



3 7 13

**Figure 6.23 Three sample images from operator 50's attempt at the third task. The triangles in the first two images are due to the reflection of the sun. It was nearly impossible to see any features in these images. Operator 50 did not notice the "car dealership flags" on the left hand side of image 13.**

In fact, the truck on the aeromap was not to scale. The aeromap was limited to a fairly small size because the amount of space available on the laptop was so limited. To make the aeromap more useful, the scale of the map itself was made to cover a large area. But when the vehicle was drawn to scale on the map, it was very small, and this made it difficult to determine the vehicle's orientation. So a decision was made to keep the aeromap scale the same, but to draw a larger vehicle. Pretesting of the system showed that this combination worked well. Operators used the aeromap to approach the goal. As they came near to the goal, however, operators relied on the image transmitted from the vehicle, and used the aeromap only to confirm that the goal was indeed the goal.

This worked well, but depended on the operator recognizing the goal in the image as the vehicle became closer to the goal. In 50's case, however, the wide field of view lens made objects appear much smaller. This fact, combined with the lack of an automatic iris control, meant that 50 did not make it to the goal.

The longest successful run was made by operator 40, whose images are shown in Figures 6.24 and 6.25. 40 noticed what looked like a path in the first few images, and spent a lot of time searching around for that path. In fact what 40 noticed was an area of ground that had no vegetation, in an area which was mostly covered with weeds. It was this hunt for a path that caused 40 to direct the vehicle first to the right (the correct direction to the goal, and then to the left in image 3.

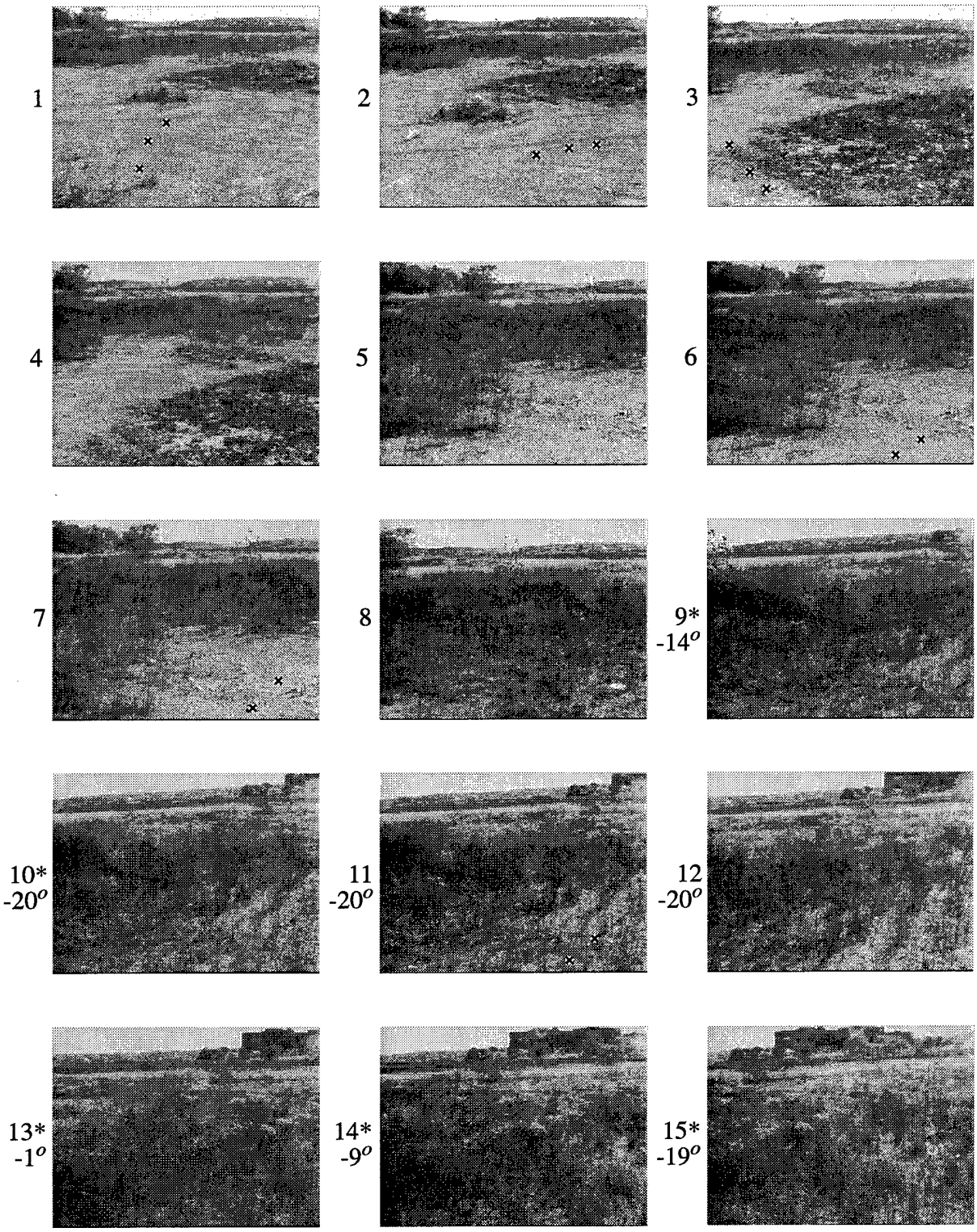



**Figure 6.24 The first 15 images and points picked in the third task by operator 40, the operator who travelled the furthest distance but still correctly completed the task. Tilt = -12 for all, pan = 0 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans the camera to the left.**

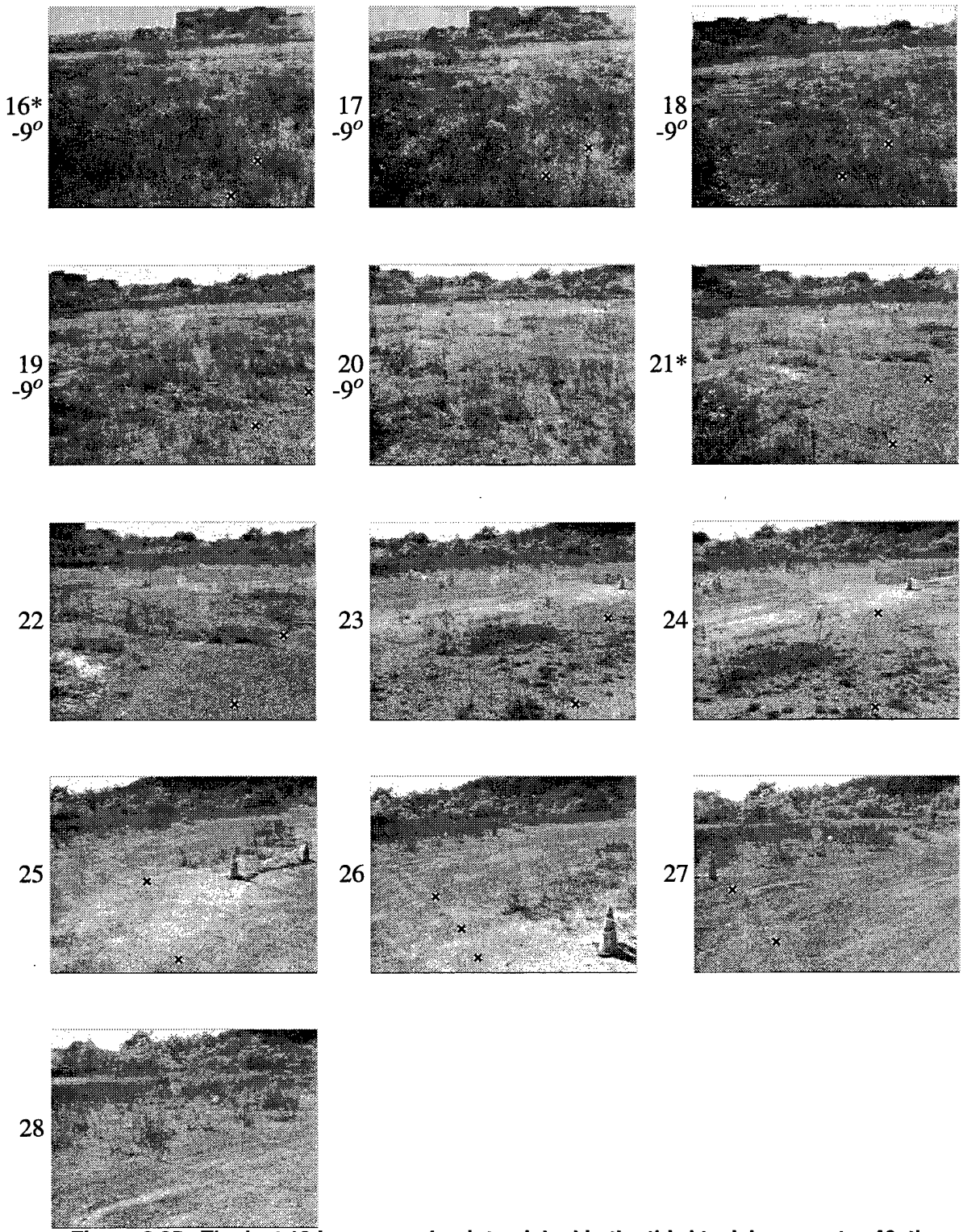



**Figure 6.25 The last 13 images and points picked in the third task by operator 40, the operator who travelled the furthest distance but still correctly completed the task. Tilt = -12 for all, pan = 0 for all except as noted below image number. A * indicates a change in pan from the previous image. A positive value pans the camera to the left.**

Images 8 to 11, and 12 to 16 were spent panning the camera in search for the apparent path which had been lost. Finally 40 realizes there is no path and uses the aeromap to find the goal.

### 6.3.3.2 Operators' Reactions to the Task

The initial yaw of the vehicle determined the initial orientation of the vehicle drawing in the aeromap. Because of the layout of the course and the fact that the inertial sensor usually initialized a yaw of 0 to be North, the initial yaw of the vehicle was usually about 240 degrees. This meant that the first aeromap typically looked like Figure 6.26.

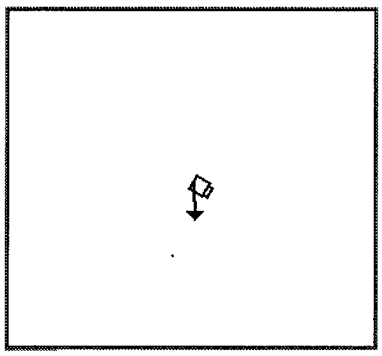


**Figure 6.26 The first aeromap operators typically saw at the beginning of task 3.**

This was in contrast to the sample map that participants were shown before the task (Figure 6.16) which had the vehicle initially pointing upward, and confused some of the operators.

> *40: I was very disoriented [in the third task] when interpreting the position of the vehicle. So much that I forgot the task.*

Some operators initially thought that they had to turn the vehicle around so that it was pointing upwards in the aeromap before they could reach the goal. Consider the first image and points picked by operator 33, shown in Figure 6.27. After seeing an updated aeromap image with the new position of the vehicle relative to the old one 33 recognized the mistake and went on to have the fastest task 3 run.

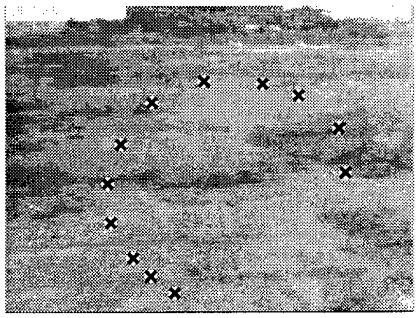


**Figure 6.27 Operators 33 tries to turn the vehicle around after seeing the first aeromap.**

Several operators liked the aeromap interface once they got the hang of it:

*44: The map was nice ... because it gave more feedback about where the vehicle was.*

*24: The last task seemed surprisingly simple compared to the second. I was surprised that the teleoperation delay didn't seem to be a factor.*

*31: The aeromap was the most fun.*

30 found it both confusing and useful.

*30: The aeromap orientation was initially confusing.*

*30: [In the third task] the aeromap was useful.*

## 6.4 Analysis of User Performance

While the tables of data give us significant insight into how operators performed on the various tests, they do not, for the most part, explain why certain operators performed better than others or what motivated operators to perform a task a certain way. To better understand what operators were doing, careful notes were taken of their comments as they performed the various tasks. These comments provide a tremendous amount of information that explains why operators did what they did, and how the interface could be changed to improve operator performance.

In addition to their comments, operators in the study performed the tasks in ways that had not been previously attempted, and provided more insight into the working of the system. This section contains an analysis of the user performance, section 6.5 contains my own view of the changes that should be made to the system and to user training based on this analysis.

### 6.4.1 Images and Point Picking

#### 6.4.1.1 Image Digitizing and a Moving Vehicle

By far, the most common misunderstanding among operators when they began to use the system had to do with when new images are digitized.

As was described in Section 3.2, STRIPE is designed to continue moving as long as it has waypoints ahead of the vehicle. When a set of points is sent from the operator to the vehicle, the vehicle takes a new image, and sends that back to the operator. Meanwhile, the vehicle continues to follow the path.

The important point to note is that the image that the operator receives is often "old" by the time they receive it, i.e. the vehicle is no longer in that position. This is not a problem for the STRIPE system, points that are behind the vehicle are ignored.

This was explained to operators during their training. The following is an excerpt from the training script from appendix D:

> As soon as you send the points to the vehicle, it will start to drive and follow the points. It will also send you a new picture, you can see the message window is telling you that picture is being transmitted. Once the new picture comes, you should repeat the point picking process, until the vehicle finishes the task -- I'll tell you about the tasks later.
>
> The new picture gets taken almost as soon as your points get to the vehicle. This means that sometimes the new picture looks almost exactly the same as the old one, so don't be surprised if you have to repick some points in the new picture that you already picked in the old one. When you send new points, the old points that you sent are thrown out.

Several operators inquired about this during the training, and the paragraph was clarified for them.

*30: Does the vehicle drive to the last point you select?*

*44: When is an image taken relative to when I send points?*

*45: Why doesn't it finish points before the next image is sent?*

*51: Does it take a picture before it gets to the end of the points?*

*62: When on a path does a picture get taken?*

For whatever reason, a very large number of operators expected that an image would not be taken until the vehicle came to a stop, and expressed surprise when an image that they received did not meet their expectations:

*33: Have I gone to the end of the path [of points picked] or just a little bit like it looks like?*

*43: I guess it moved a little bit, but I thought it was going to move farther than that.*

Several people made incorrect conclusions about how the system worked based on their observations:

*34: If I put the first point farther out it goes farther.*

*43: Can I make the vehicle go further if I pick points higher up? If I pick a point beyond the line would that make it go to the line?*

*45: If I give it a long path, how far does it go?*

Of course the comments by 34 and 45 are correct, when interpreted to mean in the context "for this path," but the test administrator took them to mean "If I pick a point farther out, the vehicle will travel a longer distance before stopping and taking the next image."

It is possible that the next two paragraphs in the training script helped to add to the confusion:

> One important thing to think about when you are picking points is that the vehicle ends up pointing in the direction it was going between the last two points you picked.
>
> *Pick 5 points, last two at an angle off to the right*
> For example, in this case, the vehicle is going to end up pointing off

> to the right. Is that clear? The system will allow you to pick a single point if you want to, but remember that this means the vehicle won't know what direction you want it to be heading in when it gets to that point. OK?

Operators expected to see one thing, but saw another:

> *62: I don't understand, every time I pick the final points to be pointing right, but I end up messed up*

As they continued to use the system, several operators began to understand what was happening.

> *41: Maybe I should keep getting a new image because the first one I get seems to be an early one.*
>
> *62: I'm no longer going to be flummoxed by the image that I see, but I don't know how to determine the endpoint. It adds uncertainty, but I won't be panicked when that happens ... Now, if I lay down points here, they may be behind the vehicle, what does that do? I'm assuming it'll ignore them. One way to determine how bad this image is to get a new image and see how much it changes. WOW! WOW! No wonder I was so confused. Whoo boy, that intermediate image is almost worthless, I'm going to get a new image every time.*

62's last sentence describes a technique that at least 7 of the operators were intentionally doing by the end of the last task. Since they knew that the image that they had just received was probably not the "final" image, they would ask for a new image frequently, to get an update of where they were in the task.

One of those users was 41, and the comment from 41 above was made during the task 2 run that is illustrated in Figure 6.21.

STRIPE was carefully designed to send images while the vehicle was still in motion. It enables operators to pick points while the vehicle is in motion, which could enable the vehicle to move a longer distance in the same period of time. Suppose that the vehicle is travelling at 1 m/s, and that it takes 3 s to digitize and transmit an image, and 2 s to pick points and send them back to the vehicle. Also assume that the operator consistently picks a path that is exactly 20 meters ahead of the vehicle at the time the image was digitized. An operator receiving images while the vehicle is still in motion could move 25% farther. This is easier to understand with the aid of Figure 6.28.

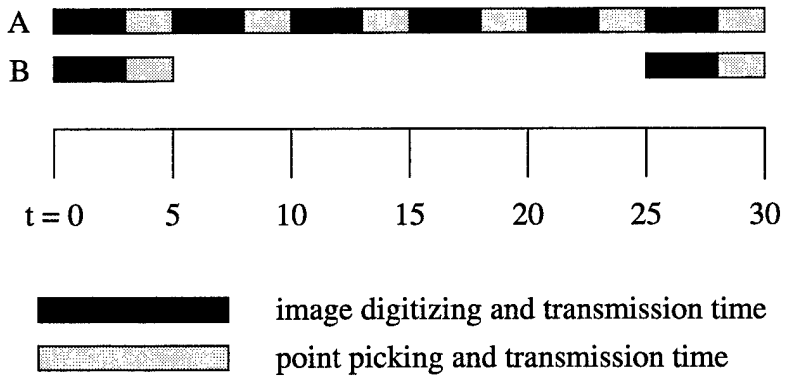



**Figure 6.28 The advantage of sending images while the vehicle is still in motion. In condition A, a new image is transmitted as soon as the points are received, even though the vehicle may still be moving. In condition B, new points are not sent until the vehicle stops moving.**

For simplicity, suppose that the vehicle is travelling in a straight line along the y axis. In time line A, the vehicle is initially stationary at time t=0 and in position y=0. As soon as the vehicle receives points, at time t= 5, it begins to move at 1 m/s, and simultaneously sends a new image (which in this case, happens to be the same image since it was taken just as movement began). The operator again picks the same points up to position y=20 in the image and sends those points to the vehicle which receives them at time 10, just as it is reaching the y = 5 mark. Another image is sent at time 10, and points up to y=25 are chosen and sent to the vehicle, reaching there at time t=15, position y = 10. The vehicle is still moving at 1m/s because it has not run out of points yet. It throws out the old points, and has 15 m of points left on the new path (having passed the first 5 m of points while the image was transmitted to the operator and points were picked). At this point, at every time t=5n+5 for integer n>2, the position will be y=5n and there will be 10 m of points left in the image that has just arrived, so the vehicle will never stop moving. In particular, at time t=30 the vehicle's position will be y=25.

Now consider time line B. Again the vehicle is initially stationary at time t=0 and in position y=0. As soon as the vehicle receives points, at time t=5, it begins to move at 1 m/s until it has completed the path at time t=25, and position y=20. It sends an image back to the operator, and sits stationary for 5s until it gets new points at t=30, when it is still at position y=20. At this point, vehicle A is already ahead of vehicle B, and since they have the same maximum speed vehicle B will never catch up. In fact, the gap will continue to increase. At every time t=5n+5 for integer

n>0 vehicle B's position will be y=4n, and so at every time t=5n+5 for integer n>2 the gap will be n meters.

#### 6.4.1.2 Image Digitizing and a Stopped Vehicle

One thing about STRIPE that had become clear before the user studies began was that the first two images in a STRIPE run always looked almost identical. As was explained in Section 3.2, this happens as follows: the vehicle is initially stationary and the first image is sent to the Operator Workstation. The operator's 2D waypoints are sent to the STRIPE Main module, and the Image Capture module is informed that it is time to digitize a new image. If the image capture module were to digitize an image immediately, the vehicle would not have moved at all, and the first two images would look identical. To prevent this from happening, STRIPE was adjusted to pause for up to two seconds if the vehicle and the camera position had not moved since the previous image was digitized.

Two seconds is not really much time, when the speed is being controlled by a safety driver who has a certain delay between hearing the "start vehicle" sound, and actually starting to move.

This duplicate image phenomenon occurs whenever the vehicle runs out of waypoints from image n, and comes to a stop at a point beyond the last point on the next set of waypoints from image n+1 that have not yet been received. Thus the STRIPE tests were designed with a very slow moving vehicle which, it was thought, would eliminate this problem.

What was not anticipated was how hesitant certain operators would be about picking points. Operators were instructed to be careful about the points chosen, the following is an excerpt from the training script:

> You can pick as many points as you like, and the vehicle will plan a smooth path between them. The most important thing to remember is that you only want to pick points that you are absolutely sure are where the vehicle should go. As you get higher up the image, things get further away and less clear. Be careful to pick the points accurately, and not too high up in the image if you can't pick a very clear point there. Remember the vehicle is going to go whereever you tell it to, and this is a fairly wide vehicle so be sure to give it some space.

As tables 6.7, 6.10, and 6.12 show, some operators picked points that were much further out than others. Also, in the slalom course, it was fairly difficult to pick points that were far out. So even though the vehicle was slowed down, the double image phenomenon still existed. The first pair of images in any sequence usually look almost identical, but in addition, this can be seen in images 13 & 14 of Figure 6.18, images 23 & 24 of Figure 6.19, images 5 & 6 and 7&8 of Figure 6.20, and images 3 & 4, 5 & 6, 7 & 8, 9 & 10, and 14 & 15 of Figure 6.22.

#### 6.4.1.3 Images From Unexpected Orientations

Several operators expressed a feeling of disorientation at some time during the tasks:

During task 1:

*40: This part is really confusing so I think I'll drive a little bit.*

During task 2:

*24: Oh geeze, I don't know where I am now.*

*61: I become disoriented so easily.*

During task 3:

*30: I lose bearing when I lose sight of the horizon.*

*40: I'm a little disoriented.*

Some operators were more explicit, and expressed the feeling that the image that they were looking at was taken from a completely different angle than they expected, and had nothing to do with the path that they had picked.

*50: [The vehicle made an] unexpected right turn when command was straight but off-center slightly.*

*62: [I was surprised by the] difficulty of predicting the results of a command; both direction of gaze and physical location - e.g., I'd expect to be looking down the road, but instead would be staring into the ditch.*

Initially it seemed as though perhaps the operators were correct, and that the vehicle had somehow driven far off course. But this was not the case, the vehicle was still on track. What had happened to produce this?

50's comment was based on the aeromap view, which unfortunately can not be accurately reproduced. However, we can look at the image sequences to better understand what was happening. Figure 6.29 shows the sequence of images that 50 was probably talking about.

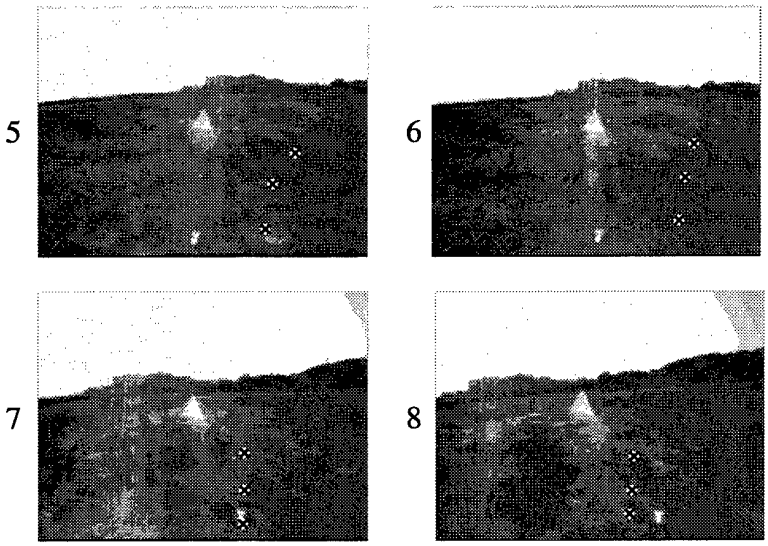



**Figure 6.29 A sequence of three images from operator 50's run. Pay attention to the points chosen and the mountains in the background.**

It is impossible to see any real features on the ground in these images, but the break between the ground and the sky in the background provides a good frame of reference. In image 5 50 starts picking points off diagonally to the right, and sends them to the vehicle. Note that images 5 & 6 are almost identical, another case of the double-image phenomenon of Section 6.4.1.2.

In image 7 the points are just as 50 described them, straight but off-center, and it is clear from the background that the vehicle has moved to the right somewhat. 50 expected that the straight line of points would cause the vehicle to keep the same orientation in image 8 as in image 7, and was surprised when it did not. One of two things happened here. The first possibility is that the points chosen in image 6 caused the vehicle to continue moving to the right while the operator was picking the points in image 7, and so image 8 was more to the right that 50 expected. The other possibility is a bit more complex.

If, however, the vehicle is near, or perhaps almost in the midst of a new set of points that have just been transmitted, the short lookahead distance (that was chosen to keep the vehicle as close to the designated path as possible, see Section 3.2), means that a much more drastic motion must take place to move the vehicle onto that new path.

Consider, for example, Figure 6.30. Imagine that the "duplicate image phenomenon" described in Section 6.4.1.2 has occurred, and so an operator picks two sets of points in what is essentially the same image. Figure 6.30 shows an overhead view of the locations on the ground of those points. The first set of points, the gray ones, are chosen and sent to the vehicle and the vehicle begins to drive. When the next, identical image arrives at the operator workstation, the user picks what appears to be approximately the same path, the clear circles. Suppose that when the new set of points arrive, the vehicle is positioned just at the second gray circle. When the clear points arrive, the vehicle is suddenly informed that it is several centimeters to the left of the path that it should be on. It makes a sharp turn to the right to get closer to the right path, and then almost immediately a correcting sharp turn to the left, see Figure 6.30. The position of the rear axle of the vehicle, the location of the origin of the vehicle, does not move too drastically. But any images digitized just as the vehicle is pulling over to the right will look as though the vehicle is headed well off the designated path. This is the second possible explanation for the vehicle apparently moving off to another location. Notice that the first two images do not have to be identical for this to happen. The problem can occur whenever the new path that the operator designates is off to one side or the other, even slightly, of the previous path.

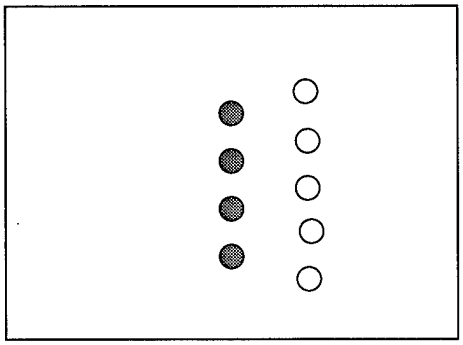


**Figure 6.30 An overhead view of the real world locations of two sets of points chosen in two identical images. One set of points was chosen to the right of the other set of points.**

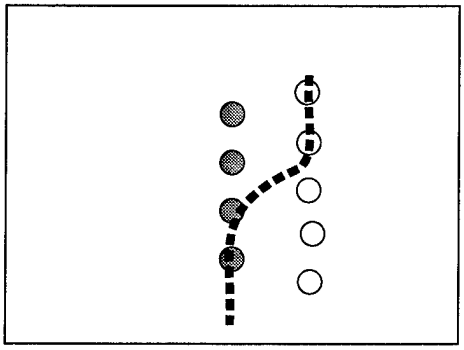


**Figure 6.31 An overhead view of the real world locations of two sets of points chosen in two identical images, and the vehicle path through those points. When the vehicle receives the second set of points, it makes a sharp turn to the right to get on to the new path.**

Sometimes operators only pick points that are very far out in the image, for example consider image 2 in Figure 6.20. Other times, operators pick points that start very close to the bottom of the screen, such as in image 10 in Figure 6.20. This problem is obviously more likely to occur when operators pick points lower in an image. If the first point on the new path is high in the image and therefore well beyond the lookahead distance of the vehicle, the vehicle will just plot a smooth path to the beginning of that path, as illustrated in Figure 6.30.

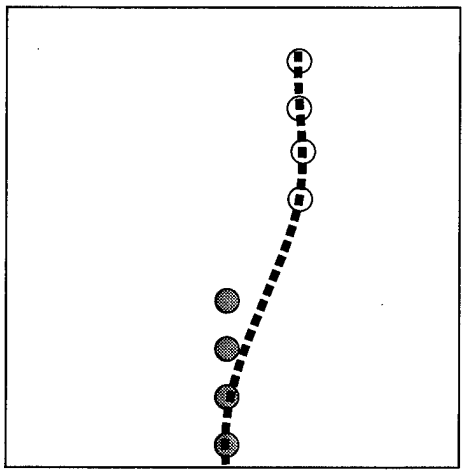


**Figure 6.32 An overhead view of the real world locations of two sets of points chosen in two identical images, and the vehicle path through those points. In this situation, the vehicle has not yet reached the new set of points, and so the correction is more gradual.**

When an operator picks a single point in an image, an additional point at the bottom center is added to make a list of two points for interpolation on the vehicle (see Section 3.2), so this makes the possibility of this sort of a correction occurring even more likely. If at least two points are designated by the operator, then no other points are added.

The obvious solution to this problem would be to reproject the points chosen in the previous image into their predicted locations in the new image. Two participants suggested just this idea:

> *50: Even better would be a cursor indicating current position on the current image, but this would be harder. This cursor should also be updated in real time, and should leave a trail.*
>
> *62: It'd be nice to regard the previous points as footprints and see them on the new image.*

An early version of STRIPE did just this. The vehicle position and orientation is transmitted back to the operator control station along with each image, and it is this information that was used to update the aeromap. It is simple to reproject the waypoints from the previous image back into the new image based on the vehicle's position and orientation information. The projected points are not necessarily accurate, as the orientation of the world ahead may have changed, but it would have been interesting to see how this extra piece of information helped or hindered users.

Unfortunately, certain elements of this interface were similar to another, patented interface, whose owner threatened us with a suit if we replicated his interface. We did not want to enter into extended legal discussions as to the validity of his claims and impede progress on STRIPE, so we removed that feature from our system. This deprived us of the opportunity to do interesting comparisons.

Finally, A few operators commented that copies of old images that had gone by would be helpful:

> *40: I was surprised at how difficult the tasks were. When the next image didn't show up as expected (didn't contain the view I predicted) I found it hard to explain, and could not recall the previous image in order to form an explanation, which resulted in not having a good overall understanding of the expected changes. Maybe being able to flip back would be useful.*
>
> *45: At the instant the image moved, I had a better sense of where the cones were.*

*61: Perhaps if we could see the previous image alongside the current image it would help prevent disorientation.*

#### 6.4.1.4 Indication of scale in the images

The images themselves contained no information about the width of the vehicle. In addition, the lack of knowledge about the shape of the terrain in front of the vehicle meant that any predictions about the width of the vehicle could be dramatically wrong. Recall the example of Section 2.4.1 where a particular point in the image was 25 meters further away from the vehicle than a flat-earth model predicted: this would also create a vehicle width in the image twice as wide as it should be.

During the operator training phase of the study, two operators asked about the width of the vehicle:

*30: Is there an indication of the width of the vehicle?*

*45: It would be useful to have an indication of the vehicle width in the image -- if I wanted to park or go around a vehicle, I want to know the width.*

In the first task, operators were told that the width of the road was about one and a half to two times the width of the vehicle, and only one operator complained about not knowing the vehicle's width. But in the second task, where operators needed to carefully navigate around the traffic cones, the lack of any information about vehicle width was a real problem:

*44: It's a little hard since I don't know how wide the vehicle is, how far out from the cones to go, but I know the width in the first task.*

*45: I don't know how much space I need to clear the cones.*

*50: It's harder for me to judge how far I am from the cones, before I just had to know what half way between was.*

Issues specific to the wide and narrow field of view lenses will be discussed in Sections 6.4.4.2 and 6.4.4.3, but several of the operators who were using the standard lens complained about the unexpected appearance of the images:

*30: The narrow field of view of about 30-40 degrees was disorienting. You forget about things outside of your field of view.* [The horizontal field of view was actually nearly 44 degrees]

*31: The cones seemed too big compared to the building behind it.* [In fact the building in the background was very far away]

*40: [I was surprised by] the scale of things in the first task.*

*41: I don't want to go too far, because I can't tell very well the depth of things in the image.*

*44: It was hard to gauge actual distances from the images.*

#### 6.4.1.5 Landmarks

Some operators made use of landmarks in the scene to help interpret an image, and to determine how far the vehicle had travelled.

*30: I lose bearing when I lose site of the horizon.*

*40: I probably won't see this cone in the next picture*

*50: Oh - there we are. [saw the cargo van in the image]*

*51: Being able to see the line move towards me lets me know how fast I'm moving.*

*62: "I want to see these cones in the next image.*

After the tests were over, 61 described how he used landmarks:

*61: First point the camera in the direction I want to go in, then move forward, but not so far that all landmarks are out of view. That way it was easier to reorient the camera....The biggest difficulty is that you can't fix the camera on an objective/ landmark throughout a move. When you turn, it's easy to become disoriented.*

The set of tasks that operators performed during our tests were fairly limited on landmarks, perhaps operator's performance would have improved in a less uniform environment. There were a lot of cones on the first course, but not a lot of ways to distinguish between them, as 51 pointed out during that task:

*51: "It would be nice if the cones had numbers on them, I'm very confused about how far I've moved"*

The second task does, to a certain extent, have more landmarks, but because in most images the cones in the distance are not clear this probably provided limited help.

#### 6.4.1.6 Poor Image Quality

At least 8 of the operators complained about the low quality of the images that they were receiving, and they were justified in doing so. The operator workstation only provided 8 different graylevels, and depending on the lighting and the environment itself, this sometimes led to very confusing images. While this research has demonstrated that operators can navigate using extremely poor images, it would be interesting to see if performance improves with better images.

One operator suggested that color would help. It was felt that the lack of color in our test site meant that the only thing color would do in these tests would be to make the orange and blue traffic cones stand out better against the background. Results discussed in Section 2.3.1.2 suggest that color is a feature that would be worth studying in future tests.

#### 6.4.1.7 Long Delay Between Images

Several operators commented on the long delay between images, especially during the periods where there was difficulty maintaining line of site between the vehicle and the operator workstation and the transfer time sometimes took minutes. 44 suggested that the system sound a tone when a new image was sent, and 40 suggested a status bar that would indicate the percentage of the image that has been transferred.

### 6.4.2 Vehicle Control

#### 6.4.2.1 Panning and Tilting the Cameras

It seemed obvious to me, the most experienced STRIPE user, that at the beginning of the third task the obvious thing to do would be to pan the camera to the right, the direction of the goal, and then to pick points. Only two operators, 31 and 45, panned before picking points.

The only other experienced STRIPE operator at Carnegie Mellon was an undergraduate who helped to test the system, layout the tasks, and do some of the user studies. In the middle of one test, while she was picking points, she did something unexpected. She wanted to turn the vehicle to the right. She panned the cameras to the right, and got a new image. Then, she panned the camera back to center and picked points. When asked about her behavior, she explained that she did the same thing that she does when she is driving her car and wants to turn at an intersection. First

she looks to the right to see the intersection. Then she looks straight ahead and turns the car into the intersection.

The training script never explicitly explained that operators were allowed to pick points in images while the camera was panned. Nevertheless, all but two operators, 34 and 51, did eventually pick points in an image that was panned.[1] However, it was not necessarily clear to them that this was legal until they tried it. Different operators seemed to work this out at different times:

During training:

> *62: I'm trying to think ... the vehicle knows how its oriented. I can sort of ignore the pan angle when choosing the path I want to take, just send it to pan in the direction I want to go and then just pick.*

During the second task:

> *36: Can I pick points even though the camera has been panned and tilted?*
>
> *44: I'm going to look to the left so I'll have a place to point to* [i.e. pick points] *to go left....I want to pick points on the left side of the screen, but the center of the vehicle is at the bottom center of the image, so I don't know if this will work ... oh ... looks like it did ok*
>
> *44: The camera is pointing to the left, so I'm looking to the left, so my inclination is to reset the camera to straight ahead, that that would work best, but it seems to work ok with it turned too.*
>
> *61: I'm just panning the image so I can get points more to the left.*

During the third task:

> *45: If I pick points while I'm panned to the right will I go to the right?*

After the tests, when asked if they had developed any sort of a system for performing the tasks, 44 elaborated:

> *44: Thinking less about where the camera was pointing and just placing points where I wanted to go (this sometimes worked better, but less so on task 2).*

1. 51 did pan the camera in task 3, and 34 panned it at the very end of task 1, but never got an image with this new pan, so this is not reflected in table 6.7.

At least two of the operators complained that panning and tilting the cameras took too much time. Perhaps this was because in order to see the new pan and tilt operators have to wait for a new image. At least one operator, 30, tried to pan during task two and had a slow response due to an unusual (for tasks 2 and 3) slow-down in the data transfer over the wireless ethernet during that task.

> *30: It takes time to pan - I get a faster response if I don't pan.*
>
> *32: I might have tried to pan the camera more to get a better view of what I was doing (looking before I leaped) but it took too long to do so.*
>
> *34: If I move the cameras I'm wasting time.*
>
> *62: The problem with the pan is that I don't want to waste time getting a new image.*
>
> *62: I don't like to use the pan because of the delay. Unless I get a totally unexpected result I'm not going to use the pan.*

#### 6.4.2.2 Operators using a Non-Zero Pan

A few operators panned the camera and then forgot that they had done so as later images came in. At some point, when an unexpected image arrived at the vehicle, the operator would check the "camera angle information window" and recognize the problem.

#### 6.4.2.3 Vehicle Agility

Some operators expressed concerns that they had no feel for how sharply they could turn the vehicle.

> *35: I still don't have a feel for tight turns. I tried to pan and then use that to go a little tighter.*
>
> *42: Would be nice to turn sharply.* [42 had experience with another teleoperated vehicle that thad the ability to make point turns]
>
> *44: The only frustrating thing was turning precisely on the slalom course, but that's probably a matter of practice rather than interface or system.*

Guidelines indicating where the vehicle could go could be superimposed on the image, but as in Section 6.4.1.4, there could be problems with accuracy on non-planar terrain.

#### 6.4.2.4 Reversing

Several operators expressed an interest in being able to reverse the vehicle:

*30: I wanted to back up -- just inverting the path the vehicle took would have been fine.*

*35: Too bad we don't have reverse.*

*36: Can I backup?*

*40: I guess you can't back it up.*

*41: Can the vehicle drive backwards?*

*42: Would be nice to backup.*

*62: If it's a single cone we're ok, if it's a double we're in deep doo-doo and I wish I could backup.*

Assuming that shifting the vehicle into reverse is not a problem, there is no reason why STRIPE could not be extended to have the capability to reverse along a path it has previously traversed. Or alternatively, an additional camera could be mounted on the rear of the vehicle, and specific points behind the vehicle could be picked.

#### 6.4.2.5 Speed Control

STRIPE was designed to drive at a constant speed. There seemed no reason to designate a particular speed for any given set of points: an operator was either confident about those points or not. However some of the participants in the study disagreed:

*32: Acceleration and deceleration controls would be a good addition. Sometimes I wanted to go slow, and others I was open to going more quickly.*

*50: It would have worked better if I also had a speed control so that I could do small frequent updates at lower speed.*

In addition, operators complained that they had little feel for how quickly the vehicle was moving:

*23: I know it shows me moving but I just don't buy it.*

*32: [I'm] not sure if the vehicle is moving or not.*

*50: I don't really have any sort of sense how fast we're moving, which makes it a bit harder.*

At least one operator thought that the speed varied:

*51: It's going a lot faster now.*

#### 6.4.2.6 Stopping and Starting the Vehicle

In general, the STRIPE main module on the vehicle assumes control of when the vehicle stops and when the vehicle starts, based on whether there are any points left in the path. But operators were given a "stop vehicle" button. The following is a description of this button from the training script:

> The vehicle is programmed to drive at about 2 miles per hour as long as it has any points to follow. When it runs out of points, it will stop, and wait for more. If, for some reason, you wish to stop the vehicle at any other time, you can press the "stop vehicle" button with your left mouse button. For example, if you send some bad points by mistake, you can press the "stop vehicle" button. (Notice that except for erasing and sending points, you always use the left mouse button) To start the vehicle moving again, press the "restart vehicle" button. After you hit the "restart vehicle" button, the vehicle will wait for a new set of points from you, and then start to move.

Only four of the operators with valid data made use of the "stop vehicle button" in any of the tasks. Of the 13 times it was used, eight of them were by operator 50, who had a very wide field of view and who wandered back and forth past the goal. Two of those eight were at the test administrator's request after the safety driver had stopped the vehicle and reported that it had to be repositioned to avoid driving in an unsafe area.

Despite the specific instruction during training to use the stop button if bad points were sent, 34 expressed a concern that he had sent bad points, but did not use the stop vehicle button.

#### 6.4.2.7 The "middle" of the vehicle

The training script instructed users that they should pick points where they wanted "the middle of the vehicle to go."

The origin of the STRIPE vehicle is on the ground at the center of the back axle. It was thought that operators would find this unnatural, and so the vehicle actually stopped when the

front of the vehicle had just crossed the points. 51 asked for clarification during training, and during task 1, 42 complained of not knowing where on the vehicle points were being chosen for.

## 6.4.3 The Basic Graphical User Interface

### 6.4.3.1 Get New Image Button

Many operators complained about the procedure for requesting a new image without picking points in the current one. In the initial design of STRIPE, the system was always in one of two modes. In "point picking" mode, operators would pick points, send them to the vehicle, and get a new image. To indicate the desire to leave point picking mode, the operator would send a path with zero points in it. Now the operator would be in "graphical user interface" or "gui" mode. In gui mode, the operator could adjust the pan and tilt of the camera, stop the vehicle, and make other adjustments on the gui such as changing the rate of image compression. When an operator was finished with gui mode, the "get new image from vehicle" button on the gui was pressed and the operator returned to point picking mode.

In subsequent upgrades to STRIPE, this concept of mode was, for the most part, dismissed. Operators could adjust the pan/tilt unit or the compression ratio at any time, and the adjustment would be reflected in the next image that was digitized on the vehicle. But the get new image procedure remained essentially the same. This interface, with the adjust compression ratio option hidden, was the interface used for all the user tests

The following is the procedure that operators had to use to get a new image without picking points in the current one, and is taken directly from the "cheat sheet" of reminders that they were given during training (see appendix D):

To request a new image without picking points:

1. If you have selected any points, erase them with the middle button.
2. Press the right button to send zero points to the vehicle.
3. Press the "get new image" button.

Operators argued that sending a path with zero points to the vehicle (steps 1 and 2 above) or clicking the "get new image from vehicle" button should be sufficient.

*23: Why do you have to right click, then get new image.*

*30: It look a lot of effort to get a new image without picking points.*

*41: You should be able to get a new image without having to first click the right button.*

*44: It was mildly annoying to have to send zero points before requesting a new image, especially when resetting a camera position.*

*45: I would expect that sending zero points should be enough to say get new image.*

*62: I wish I could send nothing and automatically get a new image.*

#### 6.4.3.2 The Restart Vehicle Button

The requirement to push the restart vehicle button after hitting the stop button (see Section 6.4.2.6) was confusing. A new image was not sent until the restart image button was pressed. After allowing 30 to flounder for some time wondering why a new image was not appearing, the test administrator instructed 30 to press the reset button so that the test could resume. The existence of the reset button was another part of the STRIPE early design that was unnecessary in the later versions.

#### 6.4.3.3 Adjusting the Pan and Tilt Values

Several operators suggested that the one degree increments for adjusting the pan and tilt values was overkill, as was the precision to one hundredth of one degree in the display for the current image's pan and tilt:

*45: Don't need one degree accuracy, 5 degrees would be fine.*

*45: I don't care about the .00 on the camera angle info.*

*62: It'd be nice to be able to pan by 5 degree increments.*

Operators 31 and 45 also complained that holding down an adjust pan or tilt arrow did not make it autorepeat, and 45 complained that these arrows were difficult to see.

### 6.4.4 Operator Interfaces

Because there were only two operators with valid data in each category of interface, it is not possible to show that any of the interfaces was statistically significantly better than any of the oth-

ers. We can, however, examine in more general terms how operators in each of the seven categories fared.

#### 6.4.4.1 Reduced Bandwidth

The interesting thing about the reduced bandwidth condition is that it has the possibility of helping operators to see the images they expect to see some of the time.

The fact that sending images takes more time means that the duplicate image phenomenon (Section 6.4.1.2) is more likely to occur, because the vehicle has more time to run out of points in a path. If the operator recognizes that the new image is a duplicate of the previous one, and picks about the same points in the duplicate image, then there is a good chance that the next image that the operator will see will be at the end of that path. As was mentioned in Section 6.4.1.2, both 34 and 35 displayed this in at least one of their runs.

On the other hand, operators running under reduced bandwidth conditions do not get much of a chance to update paths once the vehicle has begun to follow them. This may lead to errors if operators pick points very high in an image, where a single pixel corresponds to a large patch of ground. Finally, recall that almost all of the operators were presented, towards the end of the first task, with very low bandwidth transmissions due to reduced performance from the wireless ethernet.

It is impossible to say whether 34 and 35 performed well because of individual skill or because of something inherent in the lower bandwidth condition. Nevertheless, it is worth noting that each of them was the top performer in one of the tasks (34 successfully completed task 2, and 35 had the shortest distance for task 3).

#### 6.4.4.2 Narrow Field of View Lens

As was discussed in Section 6.2.6.2, the narrow field of view lens did not have automatic iris control, and so operators were probably hindered somewhat by dark or light images.

The narrow field of view lens, while limiting the amount the operator could see on the sides of the images, does cause objects within the field of view to appear larger and closer. It also allows

operators to be more precise in their point selection, because each pixel corresponds to a smaller patch of ground than the corresponding pixel as viewed through the standard lens.

One big problem with using a narrow field of view lens is that any rotations of the vehicle are emphasized. For example, consider Figure 6.33, the first three images that 62 saw at the beginning of task 1. An examination of the path that 62 took shows no sharp turn to the right. Rather, this is probably a case of an image being taken during a position correction as described in Section 6.4.1.3. The very narrow field of view lens means that none of the left side of the road is visible, while it might have been in the standard lens.

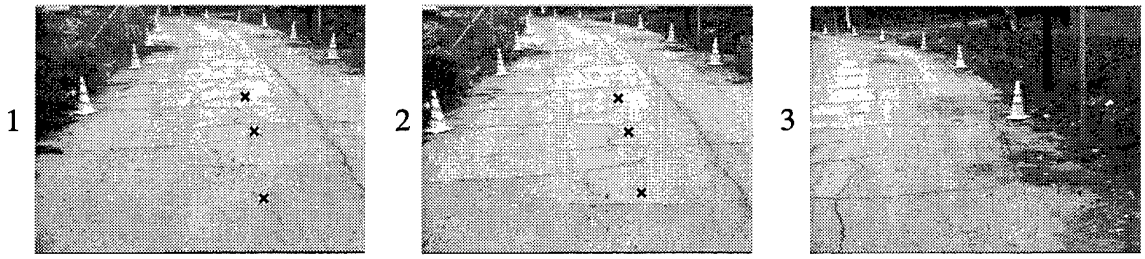



**Figure 6.33 The first three images from operator 62's first run on task 1.**

Other than 62 needing to repeat the first task, 61 and 62, the operators using the narrow field of view performed fairly well in general.

### 6.4.4.3 Wide Field of View Lens

Like the narrow field of view lens, the wide field of view lens had no automatic iris control, and so often the images were very bright or very dark. The images also were distorted at the edges, but operators did not seem to mind the distortion. In task 3, 51 picked points outside of the recommended area. Perhaps rectification of entire images taken with a wide field of view lens before displaying them is unnecessary, but rectification of those individual points picked before they are sent to the vehicle is worthwhile.

In the wide field of view lens each pixel corresponds to a larger patch of ground, so the wide field of view lens does not allow operators to pick points as precisely as the other lenses. The wide field of view lens does allow operators to see more of the context of what was going on around

them, but given the poor quality of the images, this probably does not make up for the way objects appear farther and smaller in the images.

The field of view on this lens was wide enough to include the shadow of the vehicle at times, which, changed at different vehicle orientations and different times of day. For example, Figure 6.34 contains three images taken during 51's first task. 51 recognized that this was a shadow, but was still confused that it was different at different angles:

*51: It's weird how the vehicle shadow changes.*

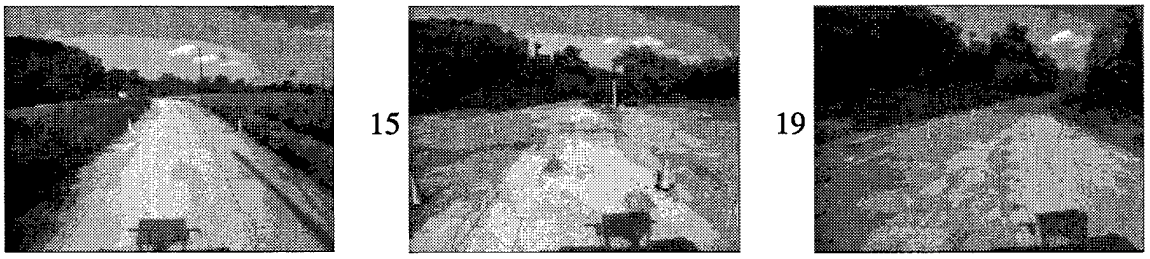



**Figure 6.34 The shadow of the vehicle was visible in some images taken with the wide field of view lens. The shadow changes as the orientation of the vehicle changes.**

#### 6.4.4.4 Maximum Compression

Because of the poor quality of the operator's monitor, the reduced quality of the images that were more heavily compressed was not as significant as it would have been on a better monitor.

33 and 36 generally did well at the tasks, though 36 did get caught by the "car dealership" obstacle in task 3.

#### 6.4.4.5 No graphical Pan/Tilt

There were two effects of removing the graphical interface to the pan tilt. First, as was expected, operators did get confused between left and right pan, though in retrospect this could have been easily remedied with left and right buttons for left and right pan instead of the up and down buttons for increase and decrease pan. 45 suggested that a graphical interface would be

helpful. Both 24 and 43 had difficulties on task 2 when they thought they were looking in one direction, but the camera was, in fact, panned in the other direction.

The other effect that was of interest was whether operators would have a hard time judging what a particular pan angle meant,e.g., how much is twenty degrees to the left?

45 generally performed tasks significantly better and faster than 31, even though they were running under the same conditions. Despite the lack of a graphical interface, 31 made more than average use of the pan, particularly in task 3.

#### 6.4.4.6 Dashboard Interface

40 and 41 generally took things more slowly than some of the other operators. It seems likely that this has more to do with their picking points relatively low in the images than with the dashboard interface. 41 was also caught by the "car dealership flags" in the last task.

The tests with the dashboard interface were held to determine whether operators found that a car-like display gave a better feel for the pan angle of the camera. One of the operators did say that the display was helpful:

*41: I like that it* [dashboard gui] *moves, it helps a lot*

During training, both operators were shown that they could move the camera pan to it's minimum extent, which moved the image graphic partially outside of the "windshield" area. The fact that the image could move to the left of the windshield was intentional, it was meant to show operators that they could pan further to the left than one could view through a standard windshield. But 41 stopped panning leftwards when the image reached the left edge of the windshield, apparently thinking that this was as far as the camera could pan.

## 6.5 Discussion

Any user study with such a limited number of participants is not designed to generate statistically significant numerical results, but rather to provide some indications of the most important effects. The quantitative performance, recorded comments, and my observation of the participants in this study provided a wealth of insight in to how the system works. This section details my view of changes that should be made to the system, and to the way users are trained.

*Further Investigate Digitizing While The Vehicle Is Still Moving*

The thing that surprised me the most about the novice users approach to STRIPE was the fact that most of the operators apparently did not understand the fact that images were digitized as the vehicle was still moving.

The operator training in this area was obviously deficient. Perhaps showing operators a real sequence of images and points picked by an expert operator with a clear explanation of the "early" images might be more helpful.

As an expert STRIPE user, I believe the early images to be helpful. At worst, I can throw these images out and get a new image. At best, I can pick more points in these images and move further faster. I would be reluctant to take this advantage away from novice users.

One possible way to get the feel for the early images across to the operators would be to reproject the previous points into the new images. Operators could be allowed to accept none, some, or all of the points (and add to the end of the path if desired). It is not obvious, however, that this will be an improvement. Operators who reject all of the previous points may forget that while they are taking the time to pick new points the vehicle is still following their old ones.

It would be worthwhile comparing novice operators using three types of interfaces:

1. STRIPE with an improved aeromap (see "Reorient Map Interface And Draw Vehicle To Scale" below) and early images.
2. STRIPE with an improved aeromap, early images, and reprojected points.
3. STRIPE with an improved aeromap, and images that are digitized only after the vehicle

reaches the end of a designated path.

I believe that with the addition of the aeromap to tell them how far they have moved, novice operators will be more comfortable receiving images taken before the vehicle has reached the end of the path, and will perform individual tasks faster than operators whose images are not digitized until the vehicle stops moving. The additional variable of reprojected points in the images should prove interesting to study.

*Correct The Duplicate Image Phenomenon*

It is possible to reduce the number of nearly-duplicate images, but not to eliminate them entirely. For this reason an addition to the script explaining that this can happen with a sample sequence of real images and points picked would be helpful.

When the Image Capture module receives a "get new image" request, and the vehicle has not moved much since the previous image was captured, it can check with the STRIPE Main module to see whether new points have been sent to the vehicle. If new points have been sent to the vehicle, the Image Capture module can continue to check whether the vehicle has made a sufficiently large move before taking an image, but the time limit at which point a new image is taken regardless of position can be extended Allowing a longer time limit will mean fewer "duplicate images," especially when the system is under manual speed control and the current shorter time limit does not take into account the slower reaction time of the safety driver.

*Reduce The Number Of Images Taken From Unexpected Orientations*

Even if operators understand that images are taken while the vehicle is still moving, they can still be surprised by images taken at unexpected orientations when the vehicle is transitioning from an old path to the current one (Section 6.4.1.3). It is possible to reduce the number of occurrences of this problem.

When the Image Capture module receives a "get new image request," again it can query the STRIPE Main module to see if new points were recently sent. The STRIPE Main module could also keep track of the vehicle yaw for the last few seconds and transmit this information to the IC

Module. The IC module could then check whether there had been a dramatic change in yaw recently, and if so wait a few seconds to allow the position correction to be made before digitizing a new image.

*Store Old Images And Allow Playback*

It would be fairly easy for the operator control station to store some number of back images and play them back for the user. This is a feature that was requested by two operators in the study, and hinted at by two others:

> *34: Maybe interpolate the frames to simulate the movement the vehicle took.*
>
> *40: Maybe being able to flip back* [through the old images] *would be useful.*
>
> *45: At the instant the image moved, I had a better sense of where the cones were.*
>
> *61: Perhaps if we could see the previous image alongside the current image it would help prevent disorientation.*

At very least operators should be able to scroll through old images. If there is not enough room on the workstation monitor to display two images side-by-side, there could be an option to display quarter-sized versions of the current image and the three images before that.

34's suggestion of interpolating the frames is an interesting one. As each new image is transmitted back to the vehicle, some extra positional data could be transmitted. This data would include 3D position data for the vehicle's path between the current image and the previous one, collected every second or every meter while the vehicle was moving. This data could be used at the operator workstation to create an interpolation between the two images using Jochem's virtual camera techniques. [19]

*Allow to pick points for left or right wheel track*

The need to know the width of the vehicle in order to closely navigate around an object on only one side of the vehicle could be eliminated by allowing the operator to choose to pick points for the left or right side of the vehicle instead of just for the center of the vehicle. This could be accomplished using either buttons on the graphical user interface or the control and alt keys on the workstation in addition to the mouse clicks. To remind the user that they were picking points in

this special mode, the design of the cursor and of the new points should have a different appearance.

This would certainly have allowed users to navigate around the slalom without needing to know the width of the vehicle, but it is not clear how comfortable novices would be with this technique.

*Improve Image Quality*

While it was encouraging to see that operators could perform the different tasks using images with only 8 graylevels, it was clear that operators disliked the low quality images. Switching to a better monitor or adding some dithering would make the operators much happier. It would probably improve the performance of operators using a wider field of view lens.

*Reduce Transmission Overhead*

It was not until tests were done on the highly compressed images that the magnitude of the overhead due to image digitization, compression, preparing for transmission, decompression, and display was recognized. It is probably possible to reduce this overhead by a few seconds by careful rewriting of some of the relevant code. This is probably a worthwhile endeavor.

The addition of a graphical indication of the percentage of the image that has already been transferred would also be welcomed by operators.

*Add Reversing Capability*

It would be fairly simple to add code that would allow the vehicle to reverse along the path that it has just traversed, and this could be very helpful to operators.

*Improve Panning and Tilting the Camera*

Operators would have a better chance at remembering that the camera was panned when the current image was taken if there were an indication in the image that this was the case. Adding a

color or patterned border around the edges of those images that were taken with a non-zero pan angle could be all that is needed to remind the operators to reset the camera pan for the next image before picking points. Alternatively, or in addition, a different cursor and point designator could be used.

Operators should also be trained to adjust the pan for the next image before picking points in the current image, to reduce the number of new images that are requested without picking points.

Operators also need to know that they can pick points while the camera is panned. The easiest way to emphasize this is to add a section to the training script in which operators pan the camera, then pick points in the panned image, and are reminded by the trainer that they are doing so.

Operators probably do not need to be reminded about the tilt of the camera, because the view of the horizon in the image is a constant indication of tilt.

*Use a Graphical Pan/Tilt Interface*

The arrows on the pan control should point left/right instead of up/down. Holding down the button on the pan or tilt control for an extended period of time should cause the button to auto-repeat more quickly. I do believe that operators may, occasionally, wish to have a finer control of the pan or tilt than 5 degrees, and I believe that an auto-repeat on the button should make operators who wanted a coarser control happy.

The dashboard interface probably does not give as much feel for angle as the compass interface, though both could be improved and compared again.

The dashboard interface should have an indication of maximum allowable pan. Perhaps something like the version in Figure 6.35, where the maximum pan is indicated by the dashed lines.

The compass interface should have the left and right pan indicated by left and right arrows, and the orientation of the vehicle used to indicate pan should be rotated as in Figure 6.36.

A comparison of these two improved interfaces is probably worthwhile.

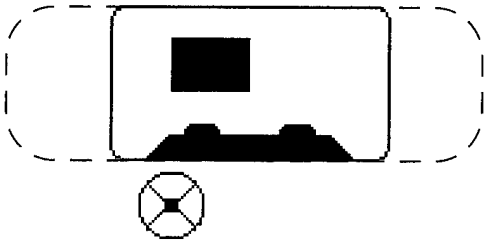


**Figure 6.35 An improved dashboard interface, with an outline indicating the allowable pan area.**

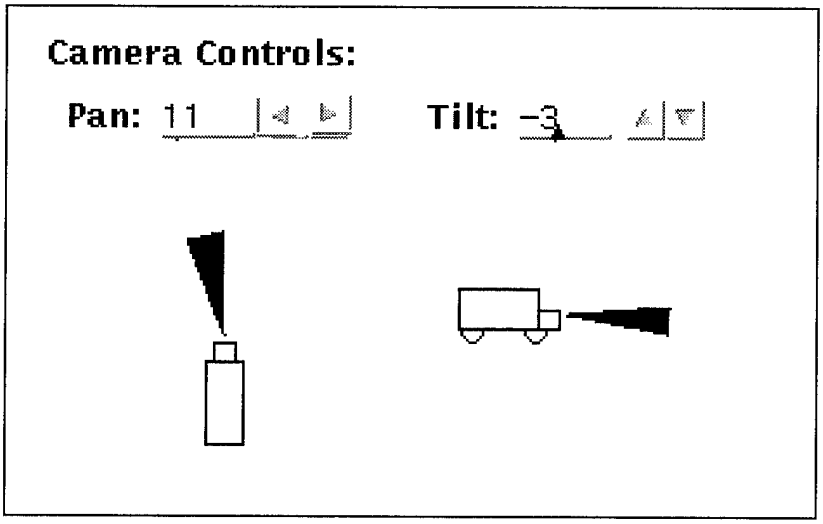



**Figure 6.36 Improved Compass Interface**

*Add Speed Control*

A simple speed control interface could indicate that the vehicle should move at a higher or lower than normal speed through certain points by allowing operators to set two speeds for vehicle movement. The standard speed would be the speed at which the vehicle would travel when points were picked in the normal manner. The alternative speed would be the speed at which the vehicle would travel when the shift key was held down as points were picked. Requiring opera-

tors to make an active change in the way they pick points (by using the shift key) might reduce the possibility that they would forget that a special speed had been chosen.

*Reorient Map Interface And Draw Vehicle To Scale*

The map interface should be adjusted so that in the first aeromap of any run the vehicle is pointing upward, as several operators indicated they found more intuitive and would prefer. This adjustment in vehicle orientation should remain constant throughout subsequent maps, even when the operator moves the vehicle off the map and the map is cleared and redrawn from a new position.

The vehicle in the aeromap was about twice the size it should have been to match the scale of the rest of the map, this should be corrected so that the vehicle is drawn to scale. The current map represents an area approximately 131 meters by 144 meters. If the vehicle were allowed to remain the same size, the area that the aeromap could cover would be quartered.

Probably the best solution is to allow the operator to zoom in and out on the aeromap. Operators can start with the vehicle at the current size and a smaller area of coverage. As the vehicle begins to move out of the covered area, operators would have the option of reducing the resolution and covering a larger area.

*Correct Get New Image From Vehicle Procedure*

The operators were right when they said that the procedure for getting a new image was too difficult. Two techniques should produce a new image on the operator's monitor:

1. The operator sends a path with zero points. No additional information is necessary.
2. The "get new image from vehicle" button is pressed. If some points have already been picked in the image a pop-up "are you sure" dialog box could appear.

*Change Stop/Reset Vehicle Button Procedure*

This was indeed confusing. The reset button should be removed. Pressing the stop button should stop the vehicle and display a dialog box informing the operator that the vehicle was

stopped and will not move again until it receives a new set of points from the operator. If an image is currently being transferred from the vehicle to the operator, the dialog box should further ask the operator if they wish to transfer a new image from the vehicles stopped position or continue transmitting the current image.

# Chapter 7 Contributions and Future Work

The STRIPE system is a unique combination of computer control and intelligent user interaction. The computer determines the 3D world path designated by the 2D waypoints and then accurately controls the vehicle over rugged terrain. The human must select points in the 2D image that accurately designate where the vehicle should go.

## 7.1 Contributions

### 7.1.1 STRIPE Works

STRIPE solves the problem of controlling a robot across a very low-bandwidth, very high-delay transmission link. Using static image data from a single camera, STRIPE enables an operator to direct the vehicle to follow a given path. STRIPE works on hilly terrain, and does not require any advance knowledge about the shape of that terrain. STRIPE's polyhedral earth reprojection techniques allow it to construct a model of the terrain in real-time, as it is needed, in order to accurately compute the necessary steering angles.

It is important to emphasize that STRIPE is specifically designed to work in situations where a few teleoperation systems would provide reduced performance, and most would not work at all.

Novice operators have used STRIPE to successfully control a remote vehicle with an 18 second delay between the time that an image was requested and the time it appeared on the operator's monitor. A semi-autonomous system requiring stereo image data would require a longer image transmission time over very low bandwidth communication links. The only other semi-autonomous system that uses monocular image data provides significantly reduced accuracy when required to follow longer paths on hilly terrain. Systems that require an operator to directly control the vehicle steering would either fail to make reasonable progress or fail to perform the required tasks with such a low image update rate. STRIPE provides the solution to the problem of vehicle teleoperation over low-bandwidth links and links with significant latencies.

### 7.1.2 STRIPE is Robust

STRIPE is tolerant to small errors in the camera calibration, camera positioning, inertial sensor, terrain prediction, and human point selection. While the Navlab 2 provided an excellent testbed for this work, STRIPE could be implemented on a much smaller and less expensive vehicle. The STRIPE inertial sensor does not require a high accuracy on a global scale. Because the system is essentially reinitialized every time a new image is digitized, it is very tolerant to drift in the global position.

An important factor in the STRIPE approach is its technological simplicity and low cost. The technology required for high-bandwidth data transmission is often very complex and expensive. STRIPE uses only simple components: a standard camera, monocular video display, mouse input, inertial sensor, and minimal computing power. The STRIPE operator workstation is simple and inexpensive, and independent from the rest of the system. It requires no knowledge about the vehicle control system or terrain, and only minimal knowledge about the camera and pan/tilt unit is necessary for the graphical display.

### 7.1.3 STRIPE Provides a Model for Semi-Autonomous Teleoperation

STRIPE provides a model of enhanced safety and proficiency for semi-autonomous teleoperation systems. By moving the computation of the trajectory onto the vehicle, a tight feedback loop can update the vehicle state information and steering commands in real-time, even if the link

between the operator and the vehicle is very slow. This enables the vehicle to make real progress, following the entire path designated by an operator in a single image.

### 7.1.4 Novice Operators Can Quickly Learn to Use STRIPE

This thesis not only describes a working system for teleoperation across low bandwidth and high latency links, it also provides important data about novice operators ability to accurately direct a vehicle under such conditions.

Operators in the user study received minimal training in the use of STRIPE. Before performing the tasks, users were only permitted to practice using the system on prerecorded images. These images were displayed in a fixed sequence; the position of a user's waypoints in the current image had no affect on the next image that was displayed. Nevertheless, 13 out of 14 operators successfully navigated the remote vehicle through the first test course the first time they used the real STRIPE system, despite the poor reproduction of images on the operator workstation monitor.

Operators were tested on two additional courses. The first, a modified slalom, was designed to start out relatively simple, and become increasingly complex. It was not expected that any of the operators would be able to complete the course. All of the operators successfully maneuvered around the first two, relatively mild obstacles on the slalom course. More than three quarters of the operators successfully completed the increasingly complex first half of the course, and one operator succeeded in completing the entire, extremely difficult, course.

Finally, operators were instructed in the use of an very limited on-line map and directed to command the vehicle to a goal that was not initially visible by travelling across nondescript terrain, and avoiding obstacles. Over two thirds of the operators successfully completed this task.

### 7.1.5 STRIPE User Studies Document Operators' Performance and Reactions

In addition to demonstrating that novice operators can use the system effectively, the user studies presented in this thesis provide a wealth of information about how users expect the system to work. The differences between operators' expectations about the system and the actual perfor-

mance of the system highlight both the inadequacies of user training, as well as the areas of the system that have potential for improvement.

### 7.1.6 A New Taxonomy For Vehicle Teleoperation

In Chapter 2, a new taxonomy for vehicle teleoperation is defined based on three variables: image update rate, transmission delay, and method of vehicle control. Using the definitions presented in that chapter, all methods of vehicle teleoperation fall into one of three categories: "Continuous and Delay Free," "Nearly-Continuous or Very-Low-Delay," and "Discrete and Delayed." This taxonomy was extremely useful in the presentation of the previous work in this thesis, and provides a good framework for future researchers in this field to classify different types of systems.

## 7.2 Recommended Changes to the Current System

The number of variables studied in the user tests made it impossible to derive results that could be formally labelled statistically significant. However, the work provides significant insight into the issues that confused novice operators, and indicates that certain changes and improvements should be made to the system. In particular, I believe the following changes to be the most important:

**Wait Until the Vehicle Has Completed the Path Before Digitizing the Next Image**

With sufficient training, operators should understand that images may be taken at intermediate points in the designated path. Nevertheless, this was by far the largest source of operator confusion about the system. Without a detailed study of different training scripts to determine how best to explain the way images are currently digitized, the best solution is to make the way the system works more intuitive to novices. This has the potential for slowing down the average speed of the vehicle. However, it is unlikely that situations in which it is critical to maintain a higher average speed will only allow for such a short operator training period.

**Be more intelligent about when to digitize an image**

By digitizing images only at the end of a path, most duplicate images and images from unexpected orientations are eliminated. If the system does allow for image digitization mid-path, it is

essential that the techniques discussed in section 6.5 are used to reduce both the number of duplicate images transmitted, as well as the number of images taken from unexpected orientations due to path correction. These two simple adjustments will provide much more useful, as well as intuitive, images to the operators.

### Store old images and allow playback

With the ability to view previous images, an operator can key in on certain landmarks in the scene and better understand how the vehicle has moved. Storing the last few images would not take up much space on the operator control station, and could provide an enormous benefit to the operators. In my analysis of user performance, I found that being able to view multiple images side-by-side helped me to better understand the movement of the vehicle. If there is space available on the operator control station to provide, in addition, reduced-sized versions of the previous images, this would probably be very helpful.

### Allow the vehicle to reverse

An additional, rear-facing camera, could be used to pick a specific path, but even simply allowing an operator to reverse along the latest path would be very useful.

### Provide an aeromap at all times

Even if there is no goal marked on the map interface, it can provide information to the operators about the distance moved and the relative orientation of the vehicle. The improvements detailed in section 6.5 should be made to the interface.

## 7.3 Future Work

There are several different extensions to the STRIPE system that are worth further investigation.

### Incorporate Safeguarded Teleoperation

The concepts of Safeguarded Teleoperation [21] could be used to extend the safety of the STRIPE system. The STRIPE operator relies on a static image to decide where to send the vehicle. It is possible that the operator might not notice an obstacle in the image (for example, a trench

in the middle of the path), or that an obstacle might move into the way of the vehicle after the image was taken.

The reprojection loop of the STRIPE Main Module has been intentionally slowed down in the current implementation in order to avoid overwhelming the vehicle's controller with position and steering requests. There are plenty of free cycles which could be used to consider whether the area ahead of the vehicle is clear and safe, or to verify that the vehicle was not on a slope that was so steep that it might topple over. The former could be as basic as bump sensors mounted on the vehicle, or a bit more advanced with the use of a simple range sensor mounted on the vehicle. The latter involves simply considering the vehicle's orientation and center of gravity.

### Provide the STRIPE Operator with Additional Information

Under low bandwidth conditions, there is a relatively low cost to providing updated position information to the operator workstation every second or so. This could be used to update the map interface and provide the operator with a better idea of the vehicle's progress.

When the bandwidth is relatively high, but there is also a high latency, there is a relatively low cost to providing additional image information for the operator, in the form of color, higher resolution grayscale images, or even additional images to provide a high resolution wide field of view.

Something else worth investigating is whether operators would find it helpful to incorporate the estimated error into the point picking process. Instead of using an arrow indicating a single pixel in the image, the cursor could be an elliptical representation of the error. The size and shape of the cursor are determined by the estimated error at the current location in the image. Error estimates could be based on precomputed values of sensor accuracy, as well as from real-time updates about the severity of the terrain based on the past several meters of travel.

When the user is picking points near the bottom of the image, the estimated error is likely to be fairly small, as points higher in the image are chosen, the estimated error will likely increase. If a user were trying to command a complex task, the error estimates might cause the operator to command the vehicle in a wider arc around an obstacle to be safe, or to pick a shorter path.

It must be noted, however, that these error estimates for point picking purposes must rely on estimates of sensor and motion accuracy along with *expected* variability in the terrain shape. If any of the actual errors are larger than the estimated bounds, the ellipses will be too small. Thus an operator picking points on an unexpectedly steep downhill segment might command the vehicle too close to an obstacle because the incorrect dimensions of the error ellipse might give the operator a false sense of confidence.

### Implement STRIPE on a Smaller Vehicle

It would be much simpler to perform further studies of the STRIPE system if it were implemented on a small vehicle that could run indoors. The use of a smaller vehicle running on indoor courses could reduce the overhead for running the user studies a tremendous amount, and enable more studies to be performed in a much shorter period of time.

### Further Investigate the Performance of Novice Operators

It should be obvious that there is significant work yet to be done to determine whether certain interfaces can be shown to be statistically better than others. Of particular interest to me is the issue of digitizing images mid-path. I would be very interested to see how the performance of novice operators carefully trained to emphasize the fact that images arrive mid-path compares with those who receive an image only when the vehicle reaches the end of the designated path.

### Investigate the Performance of Expert Operators

One of STRIPE's most natural applications, the exploration of other planets, is also one in which operators would receive significant training before attempting the real task. While a study which required the training of experts for days is impractical, one in which operators practiced real tasks for a few hours is conceivable. It would be extremely interesting to see how the skills of these operators would compare to those of the novices presented in this thesis.

# References


[1] Arps, Ronald B. and Truong, Thomas K., "Comparison of International Standards for Lossless Still Image Compression," Research Report RJ9639 (84079), IBM Research Division, Almaden Research Center, San Jose, CA, December 1993.

[2] Aviles, W.A., Hughes, T.W., Everett, H.R., Umeda, A.Y., Martin, S.W., Koyamatsu, A.H., Solorzano, M.R., Laird, R.T., and McArthur, S.P., "Issues in Mobile Robotics: The Unmanned Ground Vehicle Program Teleoperated Vehicle (TOV)", *Proceedings of Mobile Robots V*, SPIE, Boston, MA, November 1990, pp. 587-597.

[3] Byrne, Raymond H., Klarer, Paul R., and Pletta, J. Bryan, "Military Robotic Technologies," *Proceedings of the ANS Fifth Topical Meeting on Robotics and Remote Systems*, Knoxville, TN, April 1993, pp. 45-52.

[4] Cameron, Jonathan M., Cooper, Brian K., Salo, Robert A., and Wilcox, Brian H., "Fusing Global Navigation with Computer-Aided Remote Driving of Robotic Vehicles," *Mobile Robots*, Cambridge, MA, October 1986, pp. 85-89.

[5] Card, Stuart K., Moran, Thomas P., and Newell, Allen, *The Psychology of Human-Computer Interaction*, L. Erlbaum Associates, Hillsdale, NJ, 1983.

[6] Carrier, Dave, "Soviet Rover Systems," *Proceedings of the AIAA Space Programs and Technologies Conference*, Huntsville, AL, March 1992.

[7] Cooper, Brian K., Cameron, Jonathan M., Salo, Robert A., and Wilcox, Brian H., "Planetary Rover Mobility Control," *Mobile Robots*, Cambridge, MA, October 1986, pp. 330-335.

[8] DePiero, Frederick W., Noell, Timothy E., and Gee, Timothy F., "A Video Transmission System for Low-Bandwidth Remote Driving", *Proceedings of the ANS Fifth Topical Meeting on Robotics and Remote Systems*, Knoxville, TN, April 1993.

[9] Desai, Rajiv S., Wilcox, Brian, and Bedard, Rodger, "Robotic Vehicle Overview," *Unmanned Systems*, Vol. 10, No. 3, Summer 1992, pp. 16-21.

[10] Fong, Terrence, Pangels, Henning, Wettergreen, David, Nygren, Erik, Hine, Butler, Hontalas, Phil, and Fedor, Christopher, "Operator Interfaces and Network-Based Participation for Dante II," *SAE Technical Paper 951518, 5th International Conference on Environmental Systems*, San Diego, CA, July 1995.

[11] Gatland, Kenneth, *Robot Explorers*, Macmillan, New York, 1972.

[12] Glumm, Monica M., Kilduff, Patricia W., and Masley, Amy S., "A Study on the Effects of Lens Focal Length on Remote Driver Performance," *ARL-TR-25*, Army Research Laboratory, November 1992.

[13] Gomoll, Kathleen, "Some Techniques for Observing Users," in *The Art of Human-Computer Interface Design*, Laurel, B. (ed.) Addison-Wesley, Reading MA 1990.

[14] Gowdy, Jay, "IPT: An Object Oriented Toolkit for Interprocess Communication," Technical Report CMU-RI-TR-96-07, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1996.

[15] Hebert, Martial, Thorpe, Charles, and Stentz, Anthony, *Intelligent Unmanned Ground Vehicles*, Kluwer Academic Publishers (in press).

[16] Hightower, J.D., Smith, D.C., and Wiker, S.F., "Development of Remote Presence Technology for Teleoperator Systems," *Proceedings of the 14th Meeting of the UJNR/ MFP*, September 1986.

[17] Hine, Butler, Hontalas, Phil, Fong, Terrence, Piguet, Laurent, Nygren, Erik, and Kline, Aaron, "VEVI: A Virtual Environment Teleoperation Interface for Planetary Exploration," *presented at SAE 25th International Conference on Environmental Systems*, San Diego, CA, July 1995.

[18] *Jane's Spaceflight Directory*, Jane's, London, 1986.

[19] Jochem, Todd M., *Vision Based Tactical Driving*, Ph.D. dissertation, Technical Report CMU-RI-TR-96-14, The Robotics Institute, Carnegie Mellon University.

[20] Kress, Gary, and Almaula, Haren, "Sensorimotor Requirements for Teleoperation," *FMC Report R-6279*, FMC Corporation, Santa Clara, CA, December 1988.

[21] Krotkov, E., Simmons, R., Cozman, F., and Koenig, S., "Safeguarded Teleoperation for Lunar Rovers: From Human Factors to Field Trials," in *Proceedings of the IEEE Workshop on Planetary Rover Technology and Systems*, April 1996.

[22] Landauer, Thomas K., *The Trouble With Computers*, MIT Press, Cambridge, MA, 1995.

[23] Lescoe, Paul, Lavery, David, and Bedard, Roger, "Navigation of Military and Space Unmanned Ground Vehicles in Unstructured Terrains," *Proceedings of the Third Conference on Military Robotic Applications*, September 1991.

[24] Martin, Stephen W., and Hutchinson, Richard C., "Low Cost Design Alternatives for Head Mounted Stereoscopic Displays," *Proceedings of Three Dimensional Visualization and Display Technologies*, Los Angeles, CA, January 1989, pp. 53-58.

[25] McGovern, Douglas E., "Current Development Needs in the Control of Teleoperated Vehicles," Sandia Report SAND87-0646, Sandia National Laboratories, Albuquerque, NM, August 1987.

[26] McGovern, Douglas E., "Experiences and Results in Teleoperation of Land Vehicles", Sandia Report SAND90-0299, Sandia National Laboratories, Albuquerque, NM, April 1990.

[27] McGovern, Douglas E., "Human Interfaces in Remote Driving," Sandia Report SAND

88-0562, Sandia National Laboratories, Albuquerque, NM, March 1988.

[28] McGovern, Douglas E., and Miller, Dwight P., "Vision System Testing for Teleoperated Vehicles," Sandia Report SAND88-3123, Sandia National Laboratories, Albuquerque, NM, March 1989.

[29] Miller, Dwight P., "Distance and Clearance Perception Using Forward-Looking, Vehicular Television Systems," *Proceedings of the Human Factors Society 32nd Annual Meeting*, Anaheim, CA, October 1988, pp. 1453-1457.

[30] Miller, Dwight P., "Evaluation of Vision Systems for Teleoperated Land Vehicles," *IEEE Control Systems Magazine*, Vol. 8, No. 3, June 1988, pp. 37-41.

[31] Miller, D.P., and McGovern, D.E., "A Laboratory-Simulation Approach to the Evaluation of Vision Systems for Teleoperated Vehicles," *Proceedings of the International Symposium on Teleoperation and Control*, Bristol, England, July 1988.

[32] Noell, Timothy, Personal Conversation, Robotics and Process Systems Division, Oak Ridge National Laboratory, Oak Ridge, TN, November 1993.

[33] Pepper, Ross L., "Human Factors in Remote Vehicle Control," *Proceedings of the Human Factors Society 30th Annual Meeting*, Dayton Ohio, September/October 1986, pp.417-421.

[34] Piguet, Laurent, Fong, Terry, Hine, Butler, Hontalas, Phil, and Nygren, Erik, "VEVI: A Virtual Reality Tool for Robotic Planetary Explorations," *Virtual Reality World 95*, Stuttgart, Germany, Feb. 1995.

[35] Pomerleau, Dean A., *Neural Network Perception for Mobile Robot Guidance*, Kluwer Academic Publishing, 1993.

[36] Press, William H., Teukolsky, Saul A., Vetterling, William T., and Flannery, Brian P., *Numerical Recipes in C, Second Edition*, Cambridge University Press, NY, 1992.

[37] Rahim, Wadi, "Feedback Limited Control System on a Skid-Steer Vehicle," *Proceedings of the ANS Fifth Topical Meeting on Robotics and Remote Systems*, Knoxville TN, April 1993, pp. 37-44.

[38] Rahim, W. "Vehicle Remote Guidance With Path Control," United States Patent Number 5,155,683, October 1992.

[39] Sheridan, Thomas B., and Verplank, William L., "Human and Computer Control of Undersea Teleoperators", MIT Man-Machine Systems Laboratory Report, 1978.

[40] Thorpe, Charles E., ed., Vision and Navigation, *The Carnegie Mellon Navlab*, Kluwer Academic Publishers, Norwell, MA, 1990.

[41] Wallace, Gregory K., "The JPEG Still Picture Compression Standard," *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 30-44.

[42] Wilcox, Brian H., "Robotic Vehicles for Planetary Exploration," *Journal of Applied Intelligence*, Vol. 2, No. 2, August 1992, pp. 181-193.

[43] Wilcox, Brian H., "Vision-based Planetary Rover Navigation," *Proceedings of Visual Communications and Image Processing '90*, Lausanne, Switzerland, October 1990, pp. 1628-1633.

[44] Wilcox, Brian H., and Gennery, Donald B., "A Mars Rover for the 1990's," *Journal of the British Interplanetary Society*, Vol. 40., No. 10, October 1987, pp. 483-488.

[45] Wilcox, B.H., Gennery, D.B., and Mishkin, A.H., "Mars Rover Local Navigation and Hazard Avoidance," *Mobile Robots III*, Cambridge, MA, November 1988, pp. 72-76.

# Appendix A Calibration

## A.1 Computing the column and row factors

### A.1.1 The Basic Method

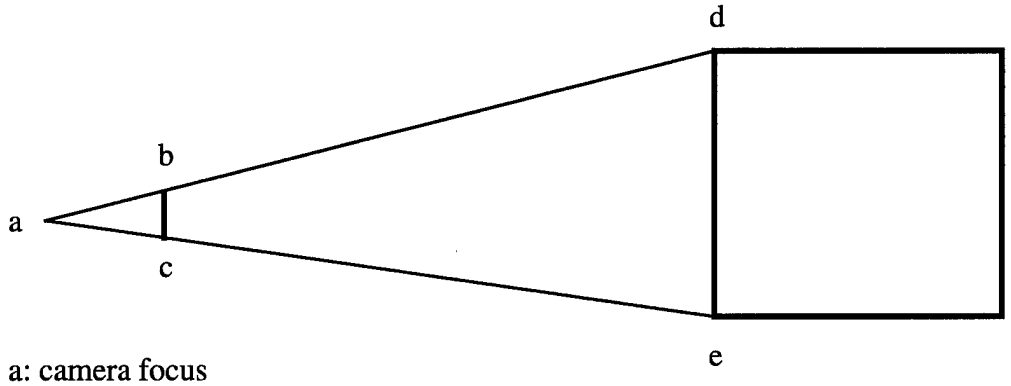
The method used for camera calibration was very straightforward and simple, and assumes a pinhole camera model. An image was taken of a square of known size, held parallel to the camera's image plane, with the top and bottom edges of the square parallel to the camera rows, and the square approximately centered in the image. This image was used to compute the camera's[1] aspect ratio:

$$\text{aspect ratio} = \frac{\text{horizontal width of square in pixels}}{\text{vertical height of square in pixels}} \tag{A.1}$$

The column factor is the distance between the centers of two pixels in adjacent columns, scaled by the focal length. This can be computed by imaging the square at a known distance from the camera and using equation (A.2), which can be derived from Figure A.1 using similar triangles.

1. Note that this is, in fact, the aspect ratio of the camera/digitizer pair, see section A.4.

$$\text{column factor} = \frac{\text{width of square in mm}}{(\text{distance from camera to square in mm}) \times (\text{width of square in pixels})} \quad \text{(A.2)}$$




a: camera focus

b-c: image of square on image plane

d-e: side of square in world

**Figure A.1 Computation of row factor**

The row factor is now computed using equation (A.3):

$$\text{row factor} = (\text{column factor}) \times (\text{aspect ratio}) \quad \text{(A.3)}$$

Note that one great advantage of this calibration scheme is that it is not necessary to know the focal length of the camera.

## A.1.2 Fine-Tuning the Values

Unfortunately this method of calibration provided row and column factors that added a bit too much error to the system, so a quick technique was developed to fine-tune the values, using the STRIPE interface. A traffic cone or other object was placed in front of the vehicle. STRIPE was configured so that it would report the 3-D location of a single point projected onto the current groundplane. The user picked the point corresponding to the intersection of a corner on the traffic cone and the floor and noted the value. The camera was then tilted down and the point corresponding to the same 3-D location was selected. If the vehicle reported that the distance to that point had increased, this indicated that the column factor was too large, a decrease in distance indicated that the column factor was too small. The column factor was manually tuned and the process was repeated. The row factor was tuned in a similar manner using the pan.

## A.2 The transformation between the vehicle and the camera

The x, y, and z location of the camera origin in vehicle coordinates was determined by careful measuring with a measuring tape and a plumb bob.

The camera was carefully aligned with the vehicle in an attempt to initialize it at a roll of zero and a yaw of zero. The initial pitch of the camera was computed by placing an object on the ground in front of the vehicle so that it appeared in the center of the image (Figure A.2). The distance between the object and the point on the ground just below origin of the vehicle was measured, as well as the height of the camera off the ground. $\tau$, the tilt angle (i.e. the pitch), was then computed as follows:

$$\tau = \text{atan}\frac{-h}{d}: \tag{A.4}$$

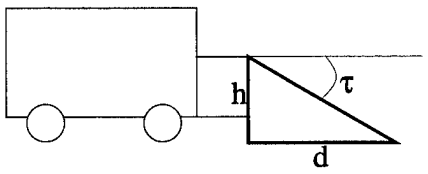



**Figure A.2 Computation of camera pitch**

## A.3 Discussion

### A.3.1 Why such a simple method?

This method of calibration is clearly not the most robust choice that one could make. Why then was it chosen? There are two strong arguments in its defense. First, it is a fairly simple procedure, and thus makes the changing to a different camera much less painful for the individual involved. Second, and most importantly, it worked fairly well in practice. The row and column factors were computed based on an object that took up most of the image, thus reducing the potential error somewhat. The x, y, and z offsets were manually measured quite roughly, but as was shown in Chapter 4, this has a negligible effect. And the initial roll, pitch, and yaw appear to be sufficiently accurate.

The method of calibration was also chosen after the elimination of other alternatives. The row and column factor of the camera could not be computed in the laboratory, because it depends on the digitizer on the vehicle (see section A.4). Calibration was attempted using a grid of points in a parking lot, but was dismissed as too inconvenient for frequent use.[2]

### A.3.2 Computation of the aspect ratio

Ideally, a sphere would make a much better calibration object than a square. When using a square to compute aspect ratio, one must be careful to align it correctly with the camera (parallel to the image plane, with the top and bottom edges parallel to the image rows, centered in the image). A sphere would only need to be placed in the center of the image.

Unfortunately, it is also important that the calibration object occupy a large portion of the image, without being held so close to the camera that it is out of focus. Manufacturing a large sphere is non-trivial, and the large sphere that was purchased for the task (a large beach ball) turned out to be surprisingly non-spherical. Since a fairly accurate square is simple to create, a square was used for computation of the aspect ratio.

## A.4 A note on camera calibration

This section is not really a necessary part of this thesis, but is being included in the hope that perhaps some roboticist who feels morally opposed to reading calibration theses might gain a little insight into camera calibration. Most of this was first explained to me by Reg Wilson.

### A.4.1 The wrong way to compute the row factor and column factor

Many people think of camera calibration as the computation of the row factor and the column factor, and would define these quantities as follows:

$$\text{row factor} = \frac{\text{(distance between the centers of two adjacent rows on the camera's image plane)}}{\text{focal length}} \qquad \text{(A.5)}$$

2. In addition to the inconvenience of having to carefully line up the vehicle with the calibration grid, there was the problem of cars parked in the lot obscuring random points on any given day, regardless of the hour.

$$\text{column factor} = \frac{\text{(distance between the centers of two adjacent columns on the camera's image plane)}}{\text{focal length}} \quad \text{(A.6)}$$

This would seem to imply that to compute these quantities, one need only consider the details of sensor size in millimeters and in pixels. In fact, the row factor and column factor are actually dependent on both the camera and the digitizer. To understand why this is, one must look at how the system works.

## A.4.2 A brief peak at the workings of cameras and digitizers

The camera output begins with a square wave to indicate the beginning of the first row, followed by an analog signal representing the image intensities along that row. This same pattern (a single square wave followed by analog data) is repeated for all subsequent rows (see Figure A.3).

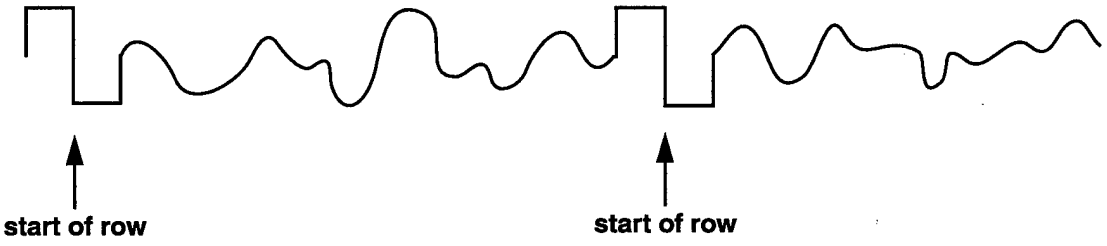



**Figure A.3 The video out signal from a camera**

The camera sensor may have the same number of pixels in a row as the digitizer, but then again it may have more or less. When the digitizer reads the camera input, the only thing that is guaranteed to be the case is that when it sees the square wave it will start a new row. The digitizer is free to determine the width of the pixels. It's perfectly possible that the digitizer will generate "narrower" pixels than the camera sensor has, and will throw out the end of the signal for each row.

## A.4.3 Correctly computing the row and column factors

The digitizer doesn't affect the "distance between row centers" computation, because it uses the camera's "start of row" signals. So, assuming the focal length is known[3], the camera specifications for sensor height in millimeters and pixels can be used to compute the row factor:

$$\text{row factor} = \frac{\text{sensor height in mm}}{(\text{sensor height in pixels}) \times (\text{focal length in mm})} \tag{A.7}$$

Determining the column factor requires the aspect ratio (equation (A.1)) and is very straightforward:

$$\text{column factor} = \frac{\text{row factor}}{\text{aspect ratio}} \tag{A.8}$$

3. Note that the focal length given in most camera specifications is given for a thin lens camera model, rather than the pinhole camera model. If you are using a pinhole camera model, you must also compute the focal length to calibrate the camera in this way.

# Appendix B Consent Form

The following page contains the consent form participants were asked to sign.

# Carnegie Mellon University

## STRIPE Vehicle Teleoperation Interface Study, Conducted by Computer Science and Robotics. Consent form

I agree to participate in experimental research conducted by members of the faculty or by students under the supervision of members of the faculty. I understand that the proposed research has been reviewed by the University's Institutional Review Board and that to the best of their ability they have determined that the observations involve no invasion of my rights of privacy, nor do they incorporate any procedure or requirements which may be found morally or ethically objectionable. If, however, at any time I wish to terminate my participation in this study I have the right to do so without penalty. I have the right to request and keep a copy of this form.

If you have any questions about this study, you should feel free to ask them now or anytime throughout the study by contacting:

Dr. Charles Thorpe
Robotics
224 Smith Hall
(412) 268-3612
cet@cs.cmu.edu

You may report any objections to the study, either orally or in writing to:

Susan Burkett
Associate Provost
Carnegie Mellon University
(412) 268-8746

Purpose of the Study: I understand I will be learning about a system for the remote control of a vehicle. I know that the researchers are studying different interfaces for this system. I realize that in the experiment I will learn how to control the system and then use the system for about an hour. I am aware that I will be videotaped during the experiment so that the researchers can find out how I use the interface.

I understand that the following procedure will be used to maintain my anonymity in analysis and publication/presentation of any results. Each participant will be assigned a number. The researchers will save the data and videotape files by participant number, not by name. Only members of the research group will view the tapes in detail.

At the end of each experiment I will be asked if I give my permission for the video tape of the experiment to be shown in public. I realize that if the video tape is shown in public, viewers may be able to identify me. I have the right to refuse this request without penalty.

I understand that in signing this consent form, I give Professor Thorpe, and his associates, permission to present this work in written and oral form without further permission from me.

_____ _____

Signature Date

_____ _____

Print Name Telephone

## Consent for public display of experimental videotape

I give my permission for the video tape of the experiment in which I have participated to be shown in public. I realize that if the video tape is shown in public, viewers may be able to identify me. I have the right to refuse this request without penalty.

_____ _____

Signature - I give permission Signature - I refuse permission

# Appendix C User Questionnaire

The following page contains the form given to the study participants before they begin the study. Subsequent pages contain the responses of the individual participants. Tests with participants 23, 24, 32, 42, and 43, had software and/or hardware failures. Empirical data about their performance was not used from their tests, though a critical incident analysis of their tests was performed, and the verbal data has been used in this thesis.

Number _____

# STRIPE Vehicle Teleoperation Interface Study Questionnaire

## General

Name: _____

Age: _____ Gender: ☐Male ☐Female

## Education

☐ Some High School ☐ High School Grad ☐ University Degree(s)

Area of Interest or Major: _____

## Driving Experience

Do you have a driver's license? ☐Yes ☐No How long have you been licensed? _____

If yes, how often do you currently drive?

☐ Never ☐ Once a year ☐ Once a month ☐ Once a week ☐ More than once a week

Do you ever do any off road driving?

☐ Never ☐ Once a year ☐ Once a month ☐ Once a week ☐ More than once a week

Have you ever driven a truck or other large vehicle? ☐ Yes ☐ No

If so, what types? _____

## Computer Experience

How often do you use a computer?

☐ Never ☐ Once a year ☐ Once a month ☐ Once a week ☐ More than once a week

What type(s) of computer do you use most frequently? _____

Do you: ☐ Like computers ☐Dislike computers ☐Feel neutral towards computers

## Video Game Experience

How often do you play video games?

☐ Never ☐ Once a year ☐ Once a month ☐ Once a week ☐ More than once a week

What types of games do you play (check all that apply)?

☐ arcade/action ☐card/gambling ☐ simulation (driving/flying) ☐adventure ☐other

## Other

Do you have any experience with remote control devices, toy or otherwise? If so please describe:

_____

Are you ☐ left-handed ☐right-handed?

Note: the word "daily" was used for the response "More than once a week"

| Operator Number | Age | Gender | Education | Major | Years Licenced to Drive | Driving Frequency | Off-Road Driving Frequency | Driven Large Vehicles |
|---|---|---|---|---|---|---|---|---|
| 30 | 27 | M | University | Electrical & Computer Engineering | 11 | daily | never | van, pickup, tractor |
| 31 | 42 | F | University | Education & Administratio n | 24 | daily | never | none |
| 33 | 38 | F | University | Mathematics | 22 | daily | never | uhaul van |
| 34 | 33 | M | University | Computer Science | 18 | daily | yearly | large van, tractor, bulldozer, backhoe |
| 35 | 29 | M | University | Cognitive Science | 13 | daily | never | moving van |
| 36 | 33 | M | University | Computer Science | 15 | daily | never | ryder 18-foot |
| 40 | 29 | F | University | Computer Science | 9 | daily | never | none |
| 41 | 28 | F | University | Physics | 10 | weekly | never | moving van |
| 44 | 30 | F | University | Computer Science | 14 | daily | yearly | none |
| 45 | 52 | M | University | Physics & Education | 35 | weekly | never | straightbody, pickup |
| 50 | 29 | M | University | Computer Science | 13 | daily | never | none |
| 51 | 20 | M | High School | Computer Science | 4 | daily | never | none |
| 61 | 27 | F | University | Chemistry | 11 | daily | never | moving van |
| 62 | 43 | M | University | Chemical Engineering | 25 | daily | never | rental moving van, small deliver (panel) truck |
| 23 | 24 | M | University | Electrical & Computer Engineering | 10 | weekly | never | pickup, minivan |
| 24 | 27 | M | University | Robotics | 11 | daily | never | 15 passenger van |
| 32 | 26 | M | University | Computer Science | 10 | daily | yearly | pickup, bus, tractor, airport-shuttle |
| 42 | 30 | M | University | Computer Science | 13 | daily | yearly | uhaul 14-foot |
| 43 | 44 | M | University | Urban Planning | (didn't respond) | daily | yearly | coca-cola truck, pickup |

Note: the word "daily" was used for the response "More than once a week"

| Operator Number | Computer Use | Computer Types | Computer Attitude | Experience with Remote Control | Dominant Hand |
|---|---|---|---|---|---|
| 30 | daily | Unix | like | toy car | L |
| 31 | daily | Mac, PC | like | toy car | R |
| 33 | daily | PC | like | none | R |
| 34 | daily | Mac, Unix | like | toy car | R |
| 35 | daily | Mac, Unix, Some PC | like | none | R |
| 36 | daily | Unix | like | none | R |
| 40 | daily | Unix, PC | neutral | none | R |
| 41 | daily | Unix | neutral | none | R |
| 44 | daily | Unix | like | none | R |
| 45 | daily | Unix, Mac, PC | like | none | R |
| 50 | daily | Unix, Mac | like | none | R |
| 51 | daily | Unix | like | toy car | R |
| 61 | daily | PC | like | none | R |
| 62 | daily | Unix, Mac, PC | like | none | R |
| 23 | daily | Unix, PC | like | toy car | L |
| 24 | daily | Unix, Mac | like | robot arm | R |
| 32 | daily | Unix, PC | like | toy car | R |
| 42 | daily | Unix | like | skid-steer vehicle | R |
| 43 | daily | PC | like | none | R |

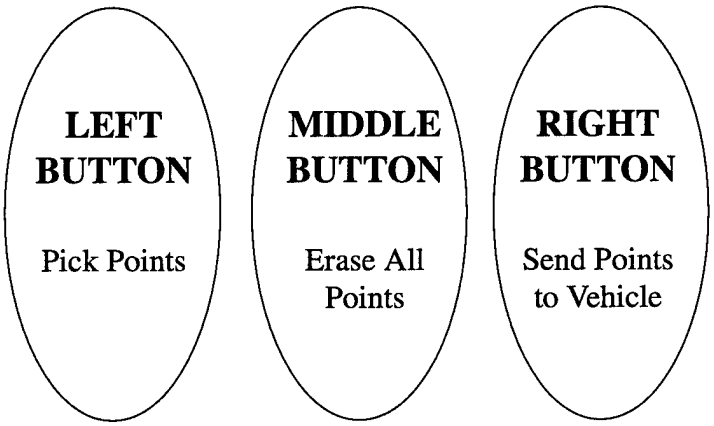
Note: the word "daily" was used for the response "More than once a week"

| Operator Number | Video Game Experience | Arcade Games | Card/ Gambling Games | Simulation (Driving/ Flying) Games | Adventure Games | Other |
|---|---|---|---|---|---|---|
| 30 | yearly | Y | N | N | N | N |
| 31 | monthly | N | Y | N | N | N |
| 33 | yearly | N | Y | N | N | N |
| 34 | yearly | Y | N | Y | N | N |
| 35 | monthly | Y | N | Y | N | Y |
| 36 | monthly | Y | N | Y | N | N |
| 40 | yearly | Y | N | Y | Y | Y |
| 41 | never | N | N | N | N | N |
| 44 | monthly | Y | Y | Y | Y | N |
| 45 | yearly | N | Y | N | N | N |
| 50 | yearly | Y | N | Y | N | N |
| 51 | monthly | Y | N | N | Y | N |
| 61 | yearly | Y | N | N | N | N |
| 62 | monthly | Y | N | Y | Y | N |
| 23 | weekly | N | N | N | N | Y |
| 24 | monthly | Y | N | N | N | N |
| 32 | weekly | Y | Y | N | N | N |
| 42 | daily | N | Y | N | Y | N |
| 43 | yearly | Y | Y | Y | Y | N |

# Appendix D Subject Training Scripts

The next page contains the "cheat sheet" given to all STRIPE users before training as an aid to remembering the things they learn in training. The following three pages contains the warping sheet (only used for the wide field of view camera), the sample slalom sheet that they are given before attempting tasks two, the airomap example sheet for task three. The subsequent pages contain the scripts used for subject training.

## Mouse commands:




## To request a new image without picking points:

1. If you have selected any points, erase them with the middle button.

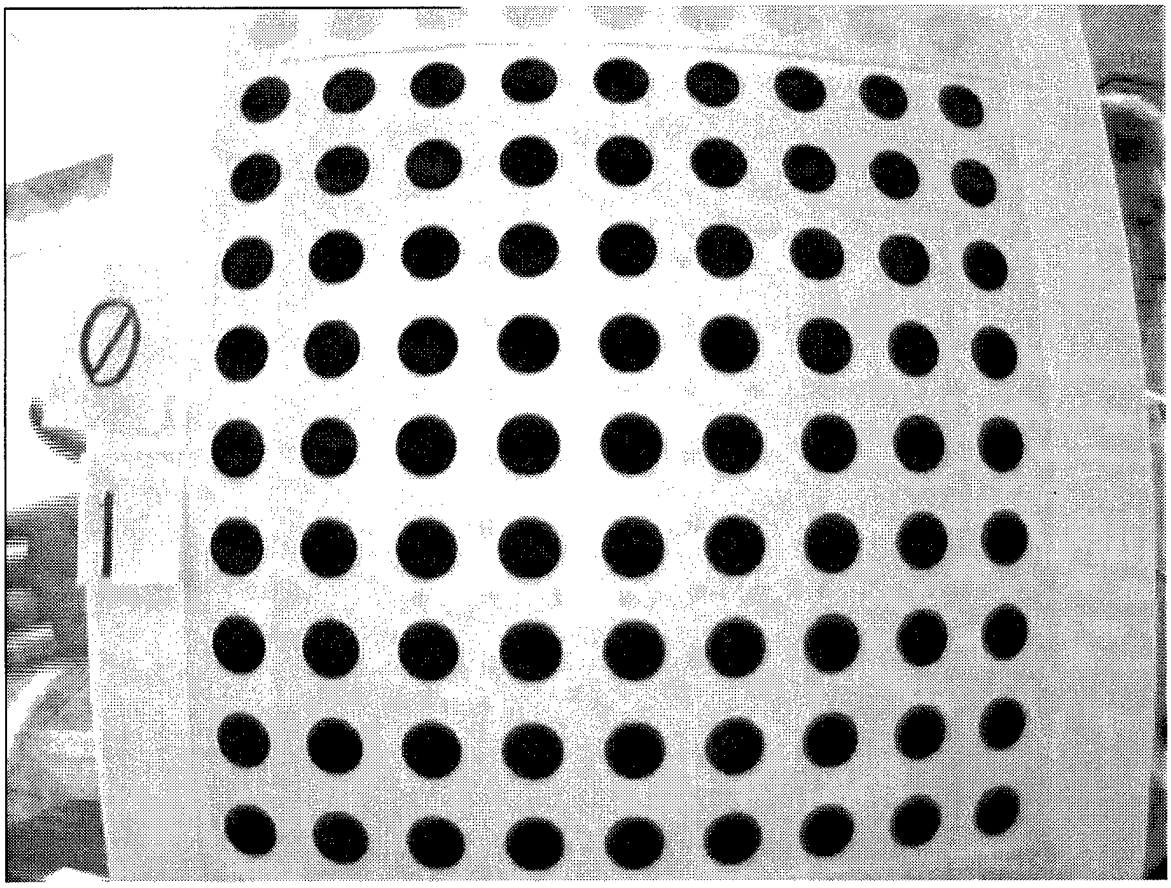
2. Press the right button to send zero points to the vehicle.

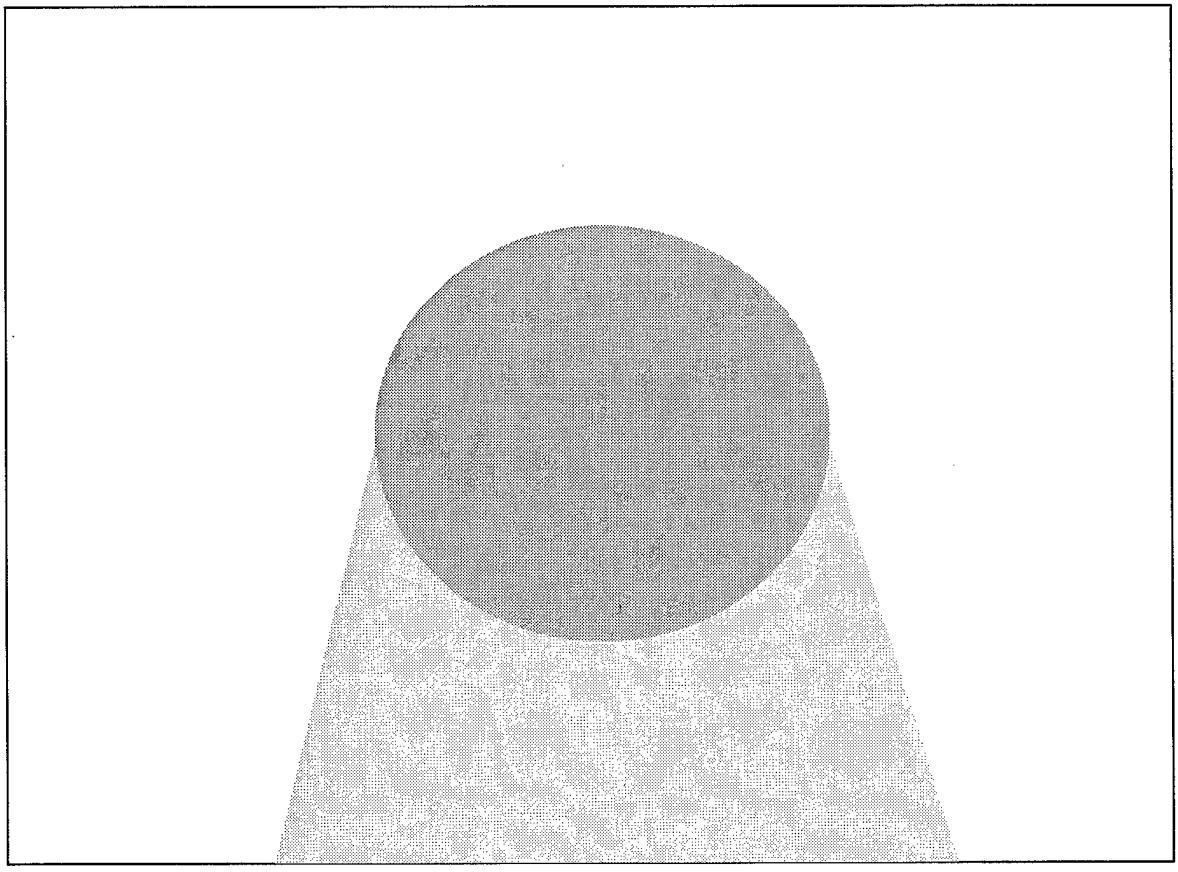
3. Press the "get new image" button.

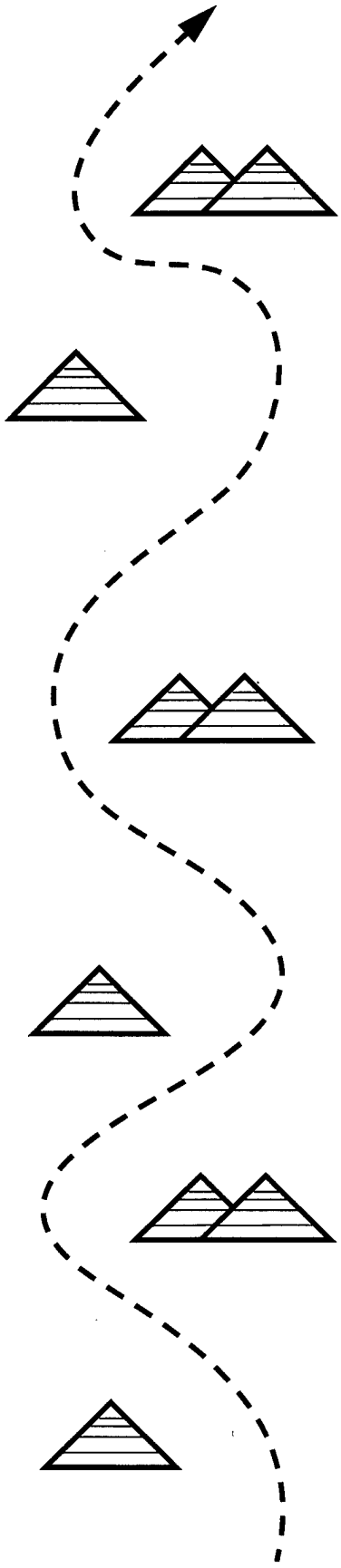
## Moving the camera:

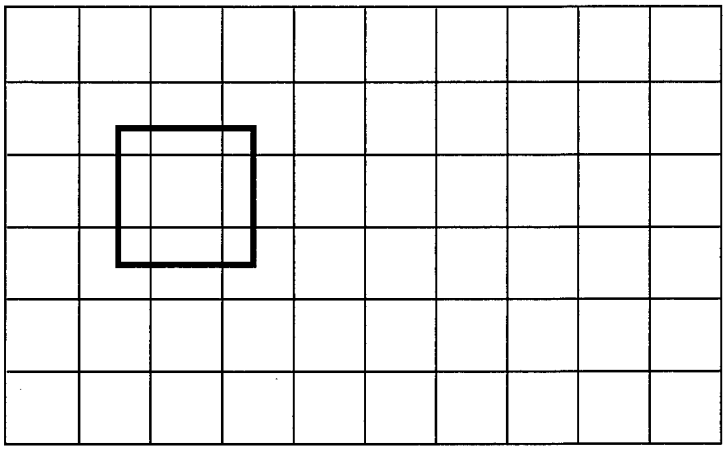
PAN: Positive=Left, Negative=Right

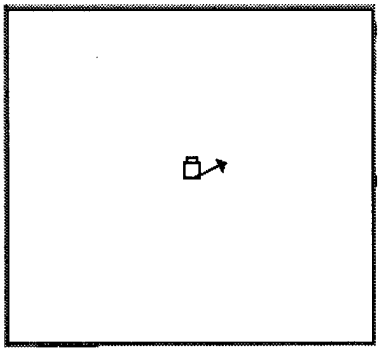
TILT: Positive=Up, Negative=Down

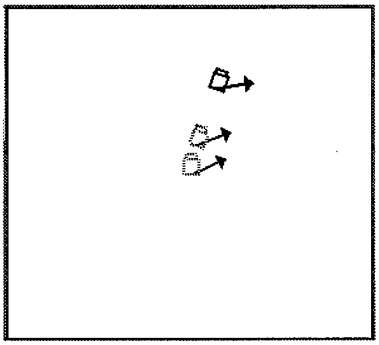


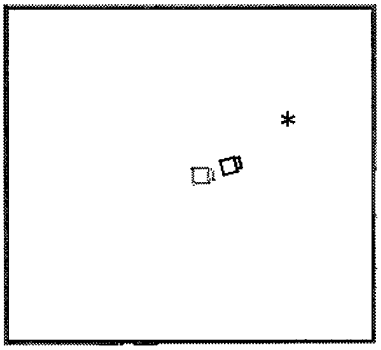












# Intro for everyone

Thanks for volunteering to participate in this study. I know it seems stupid for me to be reading this to you, but it's very important that everyone who participates in this study receives the same directions, so we'll both have to put up with it.

*Consent Form - remember to put user number on it*
We need to have your consent for you to participate in this experiment and to get your ice cream certificate. Please read this form and if you agree with what it says then sign and print your name, date, and phone number in the first section. After we have finished the study, we will ask your permission to use the video tape of your experiment in public, and you can choose to give us permission or not. If you have any questions, please ask me.

*Questionnaire - remember to put user number on it - not their name!*
Before we begin could you please fill out this questionnaire.

Two things to note before we begin. First, the point of these tests is to find problems with this system. If you have trouble with some of the tasks, it's the system's fault, not yours. Don't feel bad, this is exactly what we're looking for.

Second, remember you're a volunteer. Although we genuinely can not think of any reason why this might happen, if you become uncomfortable or find this objectionable in any way, feel free to quit at any time.

Before we start, I want you to try out the computer to play a game. Let me set that up

*Set up solitaire*

This is a card game called solitaire. Are you familiar with the rules of the game?

> *If not familiar with solitaire:*
> The object of this game is to get all the cards in the deck on these four stack areas from Ace to King, in order, by suit.
>
> *Show deck (i.e. where you get new cards)*
> This is called the deck.
>
> *Show row stacks(i.e. the "harp" where the seven stacks of cards are)*
> These are called the row stacks.
>
> You try to expose the cards that can go on the suit stacks, for instance, if you see any Aces you can put them over in the suit stack by dragging them over like this. Once an Ace is over there, you can put the two of the same suit on the Ace, and so on.

To expose more cards, you build the row stacks down in descending order alternating between red and black. Can you see the red and black colors on the screen?

*If not, tilt screen until they say yes*

*Demonstrate building up the row stacks*

If you uncover a face-down card in a row stack, you can turn it over by clicking on it.

If there's nothing you can move in the row stacks, you can turn over cards in the deck by clicking on it. Then you can use the turned-up cards too.

If you ever clear an entire row stack of cards, you can put a king or a stack starting with a king in that space.

Any questions?

You need to know is that it's very important to me that you try and think aloud as you do these exercises. It may feel a bit weird at first, but it's really very easy once you get used to it. All you have to do is say what you are thinking out loud as you work. If you forget to think aloud, I'll remind you to keep talking. I'm going to demonstrate the kind of thing I mean by playing solitaire and thinking aloud.

*Demonstrate thinking aloud*

OK, do you understand the kind of thing I'm looking for? The important thing to remember is that you try and say out loud whatever you're thinking, no matter how trivial. If you're quiet for an extended time, I may remind you to keep talking.

Now I want you to try playing solitaire while thinking aloud. Have you used a computer mouse before?

*If no mouse experience*

The mouse is correctly positioned when the buttons and the cord are away from you like this. You move the mouse around on the pad and it moves this arrow around the screen.

*demonstrate moving mouse*

see how it works? The arrow is called a cursor. If you want to move the cursor further across the screen you can lift the mouse and move it to the edge of the mouse pad, and then slide it again

*demonstrate advanced lift and slide move*

see?

The mouse has three buttons. For the solitaire game, you only ever press the left button. When you move the cursor to the area you want, and press the mouse button, it usually makes something happen on the screen. Let me click on the deck of cards

*click on deck of cards*
See - when I click on the deck of cards, it turns the next card over.

If you hold the button down instead of clicking it, it holds on to the object and lets you drag it around the screen by moving the mouse. To let go of the object, just let go of the button.

*Try and find a legal move and do it*
See, I can drag this card to another stack.

The only button on the mouse that you need to use to play this game is the left one. Now I want you to try thinking aloud while you play the game.

*Let them play and think. If they don't think aloud enough, say "keep talking"*[1]

*Quit solitaire*
Great, do you have any questions about talking aloud? OK., now I'll show you how to use the remote control system.

*Give them the "cheat sheet"*
Here is a list of the commands I will be explaining to you. I'm giving it to you now so you don't worry about forgetting what the different buttons do when I explain them to you. Don't worry about looking at it now.

Once I've explained the tasks to you I'll ask you if you have any questions about what I've explained, and I'll answer them. Once we start the task, I won't be able to provide help or answer questions. Even though I can't answer them, please ask them anyway. I'll note your questions and answer them after you're done. When you have finished all of the exercises, I'll answer any questions you still have.

1. We should have said, "*please* keep talking."

# All Point Picking Interfaces

The system that you will be testing today allows you to control a vehicle from somewhere far away. Right now all of the pictures that you're going to see are from previous tests, so don't be surprised if the next picture doesn't look the way you expect it to, they're just here for demonstration purposes.

Here's how the system works. A camera on top of the remote vehicle will take a picture and send it back to you at the computer.

*Point to ows image*
For example, here's a sample image from the vehicle.

Notice that the image looks ugly if you move the mouse outside of the image window. As soon as you move the mouse back into the window, the image will return to normal.

Your job is to tell the vehicle where to go next. You do this by using the mouse to pick a series of points in the image.

*Pick 5 points in the image*
Pick the points by using the left mouse button. The points must be chosen in the order that you want the vehicle to follow them. You pick points where you want the middle of the vehicle to go.

*Point to the STRIPE Message Window*
This is the message window. The message window will have some helpful notes that remind you which mouse button does what, in case you forget. You can also see that there is a reminder on the sheet that I gave you.

*Move the mouse in and out of image window*
Note that as you move the mouse in and out of the image window the appearance of the message window may change slightly too.

*Pick a lousy point*
Remember, we picked points in the image by using the left mouse button.

*Hit the middle button*
If you make a mistake while picking your points, press the middle mouse button to erase all the points and start again.

*Pick 3 points in the image*
You can pick as many points as you like, and the vehicle will plan a smooth path between them. The most important thing to remember is that you only want to pick points that you are absolutely sure are where the vehicle should go. As you get higher up the image, things get further away and less clear. Be careful to pick the points accurately, and not too high up in the image if you can't pick a very clear point there. Remember the vehicle is going to go whereever you tell it to, and

this is a fairly wide vehicle so be sure to give it some space.

*Press the right button*
When you are happy with the points that you have picked, press the right button to send them to the vehicle and start it moving. Be sure that you don't press the right button until you are happy with your points.

*Point to message window*
As soon as you send the points to the vehicle, it will start to drive and follow the points. It will also send you a new picture, you can see the message window is telling you that picture is being transmitted. Once the new picture comes, you should repeat the point picking process, until the vehicle finishes the task -- I'll tell you about the tasks later.

The new picture gets taken almost as soon as your points get to the vehicle. This means that sometimes the new picture looks almost exactly the same as the old one, so don't be surprised if you have to repick some points in the new picture that you already picked in the old one. When you send new points, the old points that you sent are thrown out.

One important thing to think about when you are picking points is that the vehicle ends up pointing in the direction it was going between the last two points you picked.

*Pick 5 points, last two at an angle off to the right*
For example, in this case, the vehicle is going to end up pointing off to the right. Is that clear? The system will allow you to pick a single point if you want to, but remember that this means the vehicle won't know what direction you want it to be heading in when it gets to that point. OK?

The link between you and the vehicle may be slow, and it may take several seconds before the next image is displayed. If you look at the message window, it will tell you that the image is being transmitted.

We always keep a human in the vehicle to make sure that the vehicle is safe. So don't worry, you can't break anything. The person can stop the vehicle in an emergency situation.

Before you try the real system, I want you to try the system on some prerecorded images.

# Test

- Pick a path in this image and send it to the vehicle.
- Wait for the next image, and then pick a path in it.
- Wait for the next image, and then pick a path in it.

Let me show you some more features of the system.

# For users with a wide field of view lens

*Give them the wide fov sheet*
The actual images that you will be using have been taken with a camera that has a very wide field of view. These images look like they were taken with a fish-eye lens, as you move away from the center of the image, the image warps a bit. For example, this is a picture of an image taken with the camera you will use. The dots on the sheet are really in a grid shape, but when you look at the picture, they are warped, do you see?

The vehicle has trouble with warped points, so you want to try and pick points that are not in a warped area. You should try to pick points that are within the two gray areas on this sheet. Do you understand?[2]

# All users continue here

*Display an image*
Sometimes you may feel that you don't know where to pick points in the image. Maybe the image is unclear. To get a new image without picking points in the current one, do the following.

*Pick a point or two, erase them w. the middle button, then press right button*
First, If you have picked any points, clear them out by pressing the middle button. Then, press the right button to send zero points.

*Don't press the display new image button until you have shown them the message window*
Finally, press the "get new image from vehicle" button with the left mouse button and wait for the new image to appear. You can see that when you send zero points the message window reminds you to press the "get new image from vehicle" button.

The middle of the sheet that I gave you has the procedure for requesting a new image without picking points, in case you forget.

The vehicle is programmed to drive at about 2 miles per hour as long as it has any points to follow. When it runs out of points, it will stop, and wait for more. If, for some reason, you wish to stop the vehicle at any other time, you can press the "stop vehicle" button with your left mouse button. For example, if you send some bad points by mistake, you can press the "stop vehicle" button. (Notice that except for erasing and sending points, you always use the left mouse button) To start the vehicle moving again, press the "restart vehicle" button. After you hit the "restart vehicle" button, the vehicle will wait for a new set of points from you, and then start to move.

2. Saying "do you understand" was a mistake, as it had the potential to make our users feel stupid if they said "no." It would have been better to say "do you have any questions," because individuals are used to being rewarded for asking good questions.

We always keep a human in the vehicle to make sure that the vehicle is safe. So don't worry, you can't break anything. The person can stop the vehicle in an emergency situation. I will tell you if the person in the vehicle intervenes.

Do you have any questions?

I want you to try the system on some pre-recorded images again.

# Test

- Pick a path in this image and send it to the vehicle.
- When you get the next image, pick a path in it but don't send it.
- Pretend that you made a mistake picking points. Erase all your points and pick another path. Send it to the vehicle.
- When you get the next image, pick a path in it and send it to the vehicle.
- Pretend that you don't like the next image, tell the vehicle to send a new one.
- Pick a path in the next image but don't send it.
- Pretend that you made a mistake picking points. Erase all your points and pick another path. Send it to the vehicle.
- Pretend that you don't like the next image, tell the vehicle to send a new one.

# Pan/tilt, no picture

OK, let me show you the last feature of the system. If you like, you can tell the camera on the vehicle to swivel left, right, up, and down.

*Point to Pan/Tilt control buttons. Then press them*
The left/right movement of the camera, is controlled by the pan. A pan value of 0 means that the camera is pointed straight ahead. A positive pan turns the camera to the left. So, for example, if I set the pan to positive 10, it will pan 10 degrees to the left. And a pan of negative 20 will turn the camera 20 degrees to the right.

The up/down movement of the camera is controlled by the tilt. A tilt of 0 means that the camera is horizontal. A positive tilt means that the camera is tilted up towards the sky, and a negative tilt means that the camera is pointed down towards the ground.

*Try and tilt to minimum tilt value (probably -10)*

The camera on the vehicle has a limited area that it can pan and tilt to. For example, you can see that I can't tilt the camera any higher than this.

The bottom of the sheet that I gave you reminds you which controls move what direction.

There is a delay when you send your pan and tilt commands to the vehicle. Your pan and tilt command won't take effect until the next image is taken. So if you command a pan or a tilt, and an image is currently being transmitted, your pan/tilt command won't show up until the image after that one comes in.

*Point to the camera angle info window*
It's important to look at the "Camera Angle Information" window. Remember that there is a delay, and the camera might not have made it to the angle that you moved the camera to before the current picture was taken. The "Camera Angle Information" window will tell you the angles that the camera was at when the image that you can see on the screen was taken.

# Pan/Tilt, gui picture

OK, let me show you the last feature of the system.If you like, you can tell the camera on the vehicle to swivel left, right, up, and down.

*Point to Pan/Tilt control buttons, Then press them*
The left/right movement of the camera, is controlled by the pan. A pan value of 0 means that the camera is pointed straight ahead. A positive pan turns the camera to the left. You can see the effect of your pan and tilt by looking at these pictures. The first is a picture of the vehicle from above. If I set the pan to positive 10, you can see that it will pan 10 degrees to the left. And a pan of negative 20 will turn the camera 20 degrees to the right. The black area shows the area covered by the camera[3].

The up/down movement of the camera is controlled by the tilt. A tilt of 0 means that the camera is horizontal. This picture is of the vehicle from the side. A positive tilt means that the camera is tilted up towards the sky, and a negative tilt means that the camera is pointed down towards the ground.

*Try and tilt to minimum tilt value (probably -10)*
The camera on the vehicle has a limited area that it can pan and tilt to. For example, you can see that I can't tilt the camera any higher than this.

The bottom of the sheet that I gave you reminds you which controls move what direction.

There is a delay when you send your pan and tilt commands to the vehicle. Your pan and tilt com-

3. For wide and narrow field of view users, the phrase, "though with your camera it will be wider/narrower" was added. This was probably a bad idea, since, in general, people have a difficult time ignoring things.

mand won't take effect until the next image is taken. So if you command a pan or a tilt, and an image is currently being transmitted, your tilt command won't show up until the image after that one comes in.

*Point to the camera angle info window*
It's important to look at the "Camera Angle Information" window. Remember that there is a delay, and the camera might not have made it to the angle that you moved the camera to before the current picture was taken. The "Camera Angle Information" window will tell you the angles that the camera was at when the image that you can see on the screen was taken.

# Pan/Tilt, windshield picture

OK, let me show you the last feature of the system.If you like, you can tell the camera on the vehicle to swivel left, right, up, and down.

*Point to Pan/Tilt control buttons, Then press them*
The left/right movement of the camera, is controlled by the pan. A pan value of 0 means that the camera is pointed straight ahead. A positive pan turns the camera to the left. You can see the effect of your pan by looking at this picture. It's supposed to look like the windshield of a car. The black area represents what the camera is looking at. If I set the pan to positive 10, you can see that it will pan 10 degrees to the left. And a pan of negative 20 will turn the camera 20 degrees to the right. Notice that the windshield line may go away if you pan very far to the left or to the right. This is a problem with this graphic only, just ignore it.

The up/down movement of the camera is controlled by the tilt. A tilt of 0 means that the camera is horizontal. A positive tilt means that the camera is tilted up towards the sky, and a negative tilt means that the camera is pointed down towards the ground.

There is a delay when you send your pan and tilt commands to the vehicle. Your pan and tilt command won't take effect until the next image is taken. So if you command a pan or a tilt, and an image is currently being transmitted, your tilt command won't show up until the image after that one comes in.

*Try and tilt to minimum tilt value (probably -10)*
The camera on the vehicle has a limited area that it can pan and tilt to. For example, you can see that I can't tilt the camera any higher than this.

The bottom of the sheet that I gave you reminds you which controls move what direction.

*Point to the camera angle info window*
It's important to look at the "Camera Angle Information" window. Remember that there is a delay, and the camera might not have made it to the angle that you moved the camera to before the current picture was taken. The "Camera Angle Information" window will tell you the angles that the camera was at when the image that you can see on the screen was taken.

*Change the angle, then get a new image, point to angle window.*
See, let me change the angle.... ask for a new image.... and now the new image has that angle.

# TEST (for all pan/tilt) (just do once)

Do you have any questions? I want you to try the system on some pre-recorded images one last time.

- Where do you look to see the pan and tilt values for the current image?
- What are the camera pan and tilt values for the current image?
- Pretend that you don't like this image. Set the pan and tilt so that the camera is pointing 5 degrees to the left and 10 degrees down, then ask for a new image.
- When you get the next image, pick a path in it and send it to the vehicle.
- Pretend that you don't like the next image. Set the pan and tilt so that the camera is pointing 5 degrees to the right and 20 degrees down, then ask for a new image.

# TASKS

OK, now it's time to try using the real system. There are three different types of tasks that you will be trying. Wait one minute while I set up the first task.

## Task 1: Obvious path

*Start STRIPE*
In this task your job is to drive the vehicle down a marked path. The path will be marked with cones on each side. Pay attention to the location of the cones, the path starts out on a road, but moves off of it. It's important to pick your points in the middle of the path between the cones. The width of the path is about one and a half to two times the width of the vehicle.

*Show image and point out cones*
Here's the first image from the vehicle. See the cones? At the end of the path, there is a line across the road. Your task is to command the vehicle to drive along that path, until you reach the line at the end of the path. When you think you have completed the task, let me know. OK, start the task, and remember to keep talking while you are doing it.

*If they can't do it, repeat the task up to 8 times or until they get it right.*
I'd like you to try that again. You didn't manage to successfully complete the course.

*Tell them what they did wrong.*
We're going to move the vehicle back to the start and ask you to try again.

## Task 2: Slalom

In this task your job is to drive around some traffic cones. You should send the vehicle to the right of the single cones, and to the left of the pairs of cones. The cones are spaced unevenly. You should stop when you pass the last cone or cones.

*Show overhead slalom picture*
Here's a kind of an overhead view of a similar task. Note that your cones may be spaced differently from the ones in this image. The dotted line shows the path that you would take if this were your setup. See how the vehicle goes to the right of the single cones and to the left of the double cones.

Wait one minute while I set up this task.

*Start STRIPE*
Here's the first image from the vehicle. When you think you have completed the task, let me know. OK, start the task, and remember to keep talking while you are doing it.

# Task 3: Point and Goal

This final task is a little different from the others. You are going to have a kind of a map that tells you where to go. We call this map an "airomap" Your ultimate goal is to stop at a group of three cones next to each other.

The airomap works as follows.

*Point to grid picture*
First we divide the test area up into little squares. The airomap is an overhead view of this map that takes up about one and a half squares like this bold square.

*Point to first map picture*
Here is an example of what an airomap looks like. This is the contents of that bold square. There is an overhead view of the vehicle, along with an arrow that tells you the direction to the goal. All an airomap tells you is where you are in the bold square, and what direction the goal is.

*Point to second map picture*
Every time you get a new image from the vehicle, you'll also get an update of where you are and a new arrow to the goal. The old vehicles are left on the airomap and sort of grayed out like this.

If you run off the edge of an airomap, the bold square moves over, the airomap gets cleared, and you see the next section of the airomap that has your vehicle on it.

*Point to first picture again*
So it might look like this one again.

*Point to the third map picture*
Once you get to the portion of the airomap that has the goal on it, the goal is drawn as a star on the map like this, and no arrows are drawn.

The point of this task, as I said before, is to stop at a group of three cones next to each other. Use the airomap to find the cones. Note that you might not be able to drive straight to the cones because there may be obstacles in your way. Drive to the cones in as direct a route as possible, but avoid any single cones or mounds[4] of dirt that may be in your way.

Wait one minute while I set up this task.

*Start STRIPE*
Here's the first image. And here's your airomap. When you think you have completed the task, let me know. Start the task, and remember to keep talking while you are doing it.

---

4. Several users were concerned early on about how large a mound of dirt they should avoid, so the word "mounds" was changed to "large mounds" after the first few users.

# Questions for when they are done

Was there anything that surprised you about these tasks?

Did you develop any sort of a system for doing these tasks?

Do you have any general comments about the interface, or how the system works?

Do you have any suggestions on how to improve the interface?

Was there anything that made it particularly difficult to use?

Did you have any questions about the system that you'd like me to answer now that we're done the test?

# Appendix E So You Want To Do A User Study?

## E.1 Computer Geeks: Read This Appendix!

This appendix contains some practical tips and lessons learned over the course of working on the user study portion of my thesis. If you are in computer science or robotics, and are working on a system that involves humans in some way, but haven't thought much about doing a user study, you probably want to read this appendix.

## E.2 Humans in the Loop

A lot of time and effort has been spent in the fields of computer science and robotics developing systems that work without human input. Autonomous systems usually appear to be deterministic: give them the same input and the resulting actions will be the same every time, whether the system is a robot vehicle zooming down the highway or a compiler optimizing some code. In contrast, systems that require any amount of human input often appear random upon initial inspection. Give two people the same information and they will come up with *at least* two different interpretations of that information. What do you do when you have designed a system that is intended to be used by people? What is the equivalent of leaving it in the lab for a week as it pro-

cesses example after example? You want, with some degree of confidence, to be able to evaluate your system. The only way to do this is to get some people to try it.

## E.3 Testing Your User Interface

The first decision that you need to make is how serious you are about your interface. Is this a front end that is there just to allow you to make a few changes to certain parameters in the system? Or are humans intimately involved in the performance of your system? If your answer is the latter, it's time to start to pay attention to your interface.

Once you have decided that your interface warrants some attention, you need to make a choice about how much effort you are willing to devote to the problem. You can learn a surprising amount about your system simply by getting some of your friends to try it out. By having a few people who are not intimately familiar with your work and your interface try it, you can get an idea of where some of the major problems might be. It's an easy solution that doesn't cost you much in time or effort. But it's not really very scientific, is it? And there's a lot to be said for doing something the right way.

Why be scientific? Well, for starters, you've been scientific about everything else that you have done, haven't you? You've spent a lot of time reviewing the previous work in the area, and proving how good the robot or computer side of your work is, why stop there? Don't you think that taking a scientific approach to a problem is generally a good thing anyway?

Still not convinced? Think of all the things you run into on a daily basis that have lousy interfaces. Wouldn't the world be a better place if they'd thought about it a bit more. Don't say it doesn't affect you, just because you can program your VCR. Consider the following, has this ever happened to you?

> *So I gave the clerk my credit card. She put it in the machine for verification. The machine didn't take my card. She tried again, sliding the card faster. Again, slower. Again really fast. No go. Four more tries. The she called across to the sales clerk at a nearby register. "What do I do when it won't accept the card?"*
>
> *Answer: "Enter it with the keys"*
>
> *Clerk: "I tried." Dutifully, she tried again. Once. Twice. Three times. Then she*

*called across: "Do you put in the first three numbers?"*

*"Yes."*

*"Six, three, eight?"*

*"No, four, seven, two for that machine."*

*"Oh, thanks."*

*The rest of the transaction went smoothly.* [22]

OK, so other people's systems may be crappy, but yours aren't, right? If you're not swayed by the argument to be scientific just for science's sake, maybe I can convince you that it's actually worth your while. Results that you get by asking your friends to try your system are not the kind of "Results" with a capital "R" that you can put in a paper, at least not any paper with a serious HCI audience. And if you do conduct a more formal user study, in addition to generating some real "Results," you can also be the one to establish a standard of comparison with past and future systems. Imagine: people, now and forever, setting up their tasks just like you did, so many years ago.

Hopefully by now I've convinced you that there is some value to user studies. Now I'm going to tell you what you didn't want to hear: it's going to take some significant effort on your part to do this. The purpose of this appendix is to give you a basic introduction to the kinds of things that you have to consider in order to do a scientific user study. If you decide that this is the sort of thing that you want to do, you really should go and talk to someone in the HCI Institute (if you're at Carnegie Mellon), take a course in Human-Computer Interaction, or do some more reading to get a deeper understanding of what is involved.

## E.4 Evaluating your system

Suppose you want to find out how easy it is to use your system. What do you do? Well, the first thing that you do is you come up with a definition of what you mean by easy. Is your system easy if people can accomplish a particular task in a certain amount of time? Or, is it easy if they don't make any errors while they are working on the task? Or, are you going to ask people, on a scale of 1 to 10, to rate how easy it was to use your system? Or, do you want people to try out two different systems, and tell you which was easier?

Now it's time to go back and review your motivations for doing this evaluation. What are you trying to accomplish? Do you want to show that operators perform a task using system A much more quickly than they do using system B? BEWARE! You may now be talking about proving that something is statistically significant, i.e. the probability that this would have happened by accident is very small. And usually this means testing your two systems on a lot of people. You also have to think about whether you are going to have all your users test both systems, in which case you have to start worrying about what order they do the tasks in because humans have this annoying tendency to learn while they do tasks.

If you're reading this appendix, you're probably not intending, at least at this point, to spend the next year studying your system. For goodness sake, you just wanted to know if it was easy to use, right?

OK, let's take a step back. Did you actually read my thesis? If you did, I hope that you believe that I conducted a basic user study that has a lot of value, even if the results don't have a true statistical significance. The user study portion of my thesis provided a wealth of insight into how my system works, not just into button presses on the graphical interface (though it gave me that too). I strongly encourage you to consider doing a scientific user study, but not necessarily to be pressured into doing a degree in HCI.

## E.5 Designing Your Study

The first thing that you should think about when you are designing your user study is exactly who you want to study. The answer to this question again depends on what you are trying to show. Having novice users is a good way to show that people can use your system without too much training. But is your system really designed to be used by novices? If not, you're going to have to find some users who are experts in this area already, and then take the time to ensure that they have a certain proficiency using your system.

If you do decide to use novices, be aware of their background knowledge. Do they use computers every day? Have they seen robots like yours before? Think about the kinds of things that it's important that your users know, or don't know, and be sure to recruit the right users.

Next, design the tasks that your users will be performing. Again, you want to think about exactly what it is you are trying to show, and how you are going to test it. If you're planning on having users do more than one task, think about what order those tasks should be performed in, or whether you are going to be ordering them randomly.

Now you have some writing to do. You probably want to design a questionnaire to get some demographic data on your users. You also need to write a script that you are going to use to train and test each of your users. Remember -- you want to do this in a scientific manner. You need to have a script so that all of your users receive the same instructions.

Now, go to the library and get a copy of The Art of Human-Computer Interface Design [13] and read the article by Kathleen Gomoll, "Some Techniques for Observing Users." It's only six pages long, it's an easy read, and it gives you a tremendous amount of information about what you should do when you are performing a user observation. A lot of my advice about choosing your users and designing your tasks in this appendix are based on her suggestions. So why not go and read the original.

## E.6 (The Dreaded) Human Subjects Clearance

Maybe in the back of your head somewhere, you vaguely remember hearing that when people conduct psychology experiments, or do animal testing, they have to get some bureaucratic office somewhere at their university or company to approve their tests as being humane and not being harmful. But you work with computers or robots, so none of this has anything to do with you, right? WRONG!

Guess what, you will be using humans (I presume) to do your user study. This means you probably have to get permission somewhere from someone to do these tests. The good news, is that it's probably fairly straightforward, and not too painful (though all dealings with bureaucratic machines have a certain degree of pain involved). Even though you are doing testing on humans, it's unlikely that your testing is going to panic the powers-that-be the way some weird psychology experiment might.

The best way to go about this is to find someone else who has already done it, and use their (successful) application as a template for yours. You probably have to submit a description of the experiment along with how the anonymity of your users will be protected, the consent form that users will sign, and whether your users will get any benefit from the experiment (which usually refers to the ice cream gift certificates you are bribing your users with). Here at Carnegie Mellon, the application goes through the provost's office, and is reviewed by a committee that only meets every few weeks, so it's a good idea to start early. Approval here is only good for a year, though, and then you need to ask for an extension, so don't apply *too* early. Once you've been approved, any changes you make to the information that you sent them, no matter how minor, must be run by the same office. Aren't getting any volunteers and want to offer them pizza now? Run it by the officials. Of course they'll say yes, but it keeps them happy and you ethical.

## E.7 Testing

OK, you've decided who your users will be, you've written your script, you've been approved by the mucky-mucks upstairs, you're ready to call in those users, right? Well, almost. It's a good idea before you start the real study to do some pretesting. Test one or two users, whose data you are willing to throw out, just to make sure that everything goes as you expect. There's probably something you forgot when you designed your tests or your script, and the only way to find that something is to try it out. If you try it out during a pretest, you can still change the script or your setup and not affect the outcome of your "real" experiments.

You will want to record what happens during your experiments so that you can review it later. Video taping is great, if a camcorder is available, but even just speed-writing copious notes is useful. After the experiment is over, you want to concentrate on the places where the person did the unexpected or was surprised by something they didn't expect. Consider what exactly they were trying to do. Did it work? How did they do it?

In addition to video taping or note taking you almost certainly want to automatically record what the user does within your system. Where did they click with the mouse? What keys did they type? What image was displayed on the graphical interface. This data will be a big help when you come to analyze the results.

## E.8 Practical Tips and Lessons Learned

Here are some things that you should think about when you are designing your experiments. Most of them are based on things that I did wrong.

### E.8.1 Pick a cool project

This was the one thing I did right. It makes it a lot easier to recruit users when you tell them that they're going to be remotely controlling a Hmmwv than when you tell them that they will be trying out a new spreadsheet. But you've probably defined your project already, so at least try and make it sound cool.

### E.8.2 Write a first draft of your results chapter before you have any results

I'm serious here. Before you do your main study, you want to be certain that you are recording all the keystrokes and button presses that you need. A good way to do this is to try and analyze your pretest data. You will instantly see ten things that you should have recorded but didn't. This doesn't guarantee that you won't forget something, but it certainly helps.

### E.8.3 Keep the test short

You probably don't want to be doing this test for weeks or months. Believe me, you really don't. It's fun at first, but actually doing the testing can get very old, very quickly.

When you consider how many people you can test each day, remember to include set up and clean up time in your estimate. I initially expected that it would take about an hour and a half to test each user, so I figured that four users a day would be a snap. I was wrong. First, while our test site was officially closed to the public, it was still outside, and easily accessible to anyone who cared to wander in. We did not have much trouble with people wandering in on our experiments,[1]

1. With one notable exception. Hint for graduate students: watch some TV. You should not get all of your news from National Public Radio. In particular, it's a good idea to watch the local newscast. It helps you to avoid those embarrassing encounters with public officials:

> Me: "Hi, can I help you?"
> Guy in Suit: "We're here to do an interview with KDKA TV."
> Me: "Oh, so I guess you're from the city."
> Guy in Suit: "Actually, I'm the Mayor."

but there was enough evidence that people did wander through the site[2] that we had to collect all 50 or so traffic cones each evening and set them up again in the morning. Now, while carrying cones around the slag heap did help to improve my upper body strength,[3] it took a lot of time. And the cargo van with the operator workstation setup had to be set up in the morning and packed up every night so that we could drive the van back to school, and bring the expensive computing inside. That took time. And pretesting showed that it was safer to allow a two and a half hour time slot for each user, just in case the user test itself took a bit longer, or if there was a problem with the computing. So we ended up testing two people a day. It was a lot of work to set up and clean up every day in order to test just two people. Which brings me to my next point.

### E.8.4 KISS (Keep it simple, stupid)

Actually, it's two points. First, did you notice that I said "we" a lot in the previous section? That's because in my experiment I always needed a safety driver in the robot vehicle at all times. I am forever indebted to the members of the Navlab project who spent hours sitting in the vehicle waiting for bells and buzzes to tell them to start and stop the vehicle. Not to mention helping to set up traffic cones in the morning and collect them again in the evening. But it would have made scheduling the tests easier if they hadn't had to be there. The more people that you have to coordinate to be in the same place at the same time, the more difficult things become.

Second, remember that the more hardware you use, the more there is that can break, and stop your experiments. There is a tremendous value to testing "real" systems, and not just using simulations. I am convinced that a lot of what I learned about my system would not have come out had I done the tests in simulation. But be prepared. If you're testing away from your regular lab, bring duplicates of everything you can think of. I mostly followed this advice. And over the course of my experiments, I made use of the extra ethernet cables and terminators, the extra mouse, the extra video tapes, the extra transceiver, and even the extra chair[4] that I had brought along with me. Other things that I didn't originally duplicate ended up being purchased in mid-test at Radio

2. The exploded firecracker remains kind of gave it away.
3. When I started out I could only carry four cones, by the end of the experiment, I could carry twice that.
4. I'm afraid that I'm not as athletic as some of the guys on the Navlab project. And despite my increased upper body strength due to cone-lifting workouts, I couldn't pull myself up on to the hood of the Hmmwv (to adjust the cameras) without the aid of an extra chair to help me onto the tire first.

Shack,[5] but there were still some things I couldn't duplicate. I didn't have an extra autonomous vehicle, for example, and so when the fuel pump on the generator went, I was stuck until it was repaired. If you are testing a real system, keep it as basic as you possibly can. And bring duplicates of everything you can afford. Speaking of money...

## E.8.5 Think before you skimp

*"We'll find something new to do now.*
*Here is lots of new blue goo now.*
*New goo. Blue goo. Gooey. Gooey.*
*Blue goo. New goo. Gluey. Gluey."*
*-- Dr. Seuss, "Fox in Socks"*

It's good to save money. It's bad to waste money. But when you are considering a purchase, and there is a vast difference in price between two apparently similar items, look into it closely and ask yourself why this might be the case.

I learned this lesson the hard way. I wanted to add some blue stripes on to my orange traffic cones, to make them stand out better. I looked in an industrial products catalog, and there were two types of blue adhesive tape. The "general purpose box sealing tape" cost about $5.00 a roll. The "vinyl lane marking tape" cost about $20.00 for a shorter roll. I went with the cheap stuff.

A week or two later, the cones seemed to be leaking blue goo. Worse, it was blue goo that, once in contact with your clothing, would become a permanent part of your wardrobe. It took a while to get off of your skin too. It appears that "blue general purpose box sealing tape" is, in fact, "clear general purpose box sealing tape" with blue glue. Attach it to an oily cone (the cones are oiled slightly at the factory so they don't stick to one another), leave the cone outside in the sun for several days, and you have blue glue goo.

Worse still, the blue glue, while easy to remove in small quantities,[6] is harder to remove completely off of a traffic cone, especially off of the inside of the cone which acquired the goo when stacked up with other cones.

---

5. By my advisor, no less. You know it's time to graduate when your advisor will drive out to a Radio Shack to get you a part so that you don't have to scrap yet another user test. Thanks, Chuck!

6. Particularly with a T-shirt or jeans.

Many hours of experimentation with different techniques to remove glue and goo finally paid off, but it was not a pleasant experience.[7] And I ended up buying the expensive tape in the end anyway.

### E.8.6 Pick a pleasant testing environment

Gomoll recommends that, "An ideal setting for user observation is a quiet, enclosed room with a desk." [13] I would recommend that you at least try and stay away from slag heaps. A single day of testing on a slag heap when the thermometer has gone above $90^{o}$ is sufficient to make even the most enthusiastic researcher despair. Of course, when the temperature is high, you sweat more, making the lack of bathrooms on site much less important.

But seriously, it makes your life much more convenient if you can work indoors, or, if you absolutely have to work outdoors, at least try and work near your office. The convenience of nearby bathrooms and water fountains is a big bonus. As is a place to run and hide when it rains. Also, if your users are coming from nearby, it makes it much easier if you can call them in their office and ask them to come 15 minutes later than when they are waiting somewhere outside to meet a car you have scheduled to pick them up and bring them to the site 15 minutes away. Working indoors may also have the additional benefit that you can leave your equipment set up overnight, which, as I said before, can save huge amounts of time.

## E.9 Do it!

There's no question that there is some work involved in doing a user study. But there is an awful lot of benefit that you can gain. The only way to *really* learn about how well your system works is to test it out on users. And if you've decided to do that, take the time to do it right.

7. By the way, "Goo Gone," despite its exaggerations in the area of pleasant citrus scents, is a magical product and removes even the nastiest blue goo.